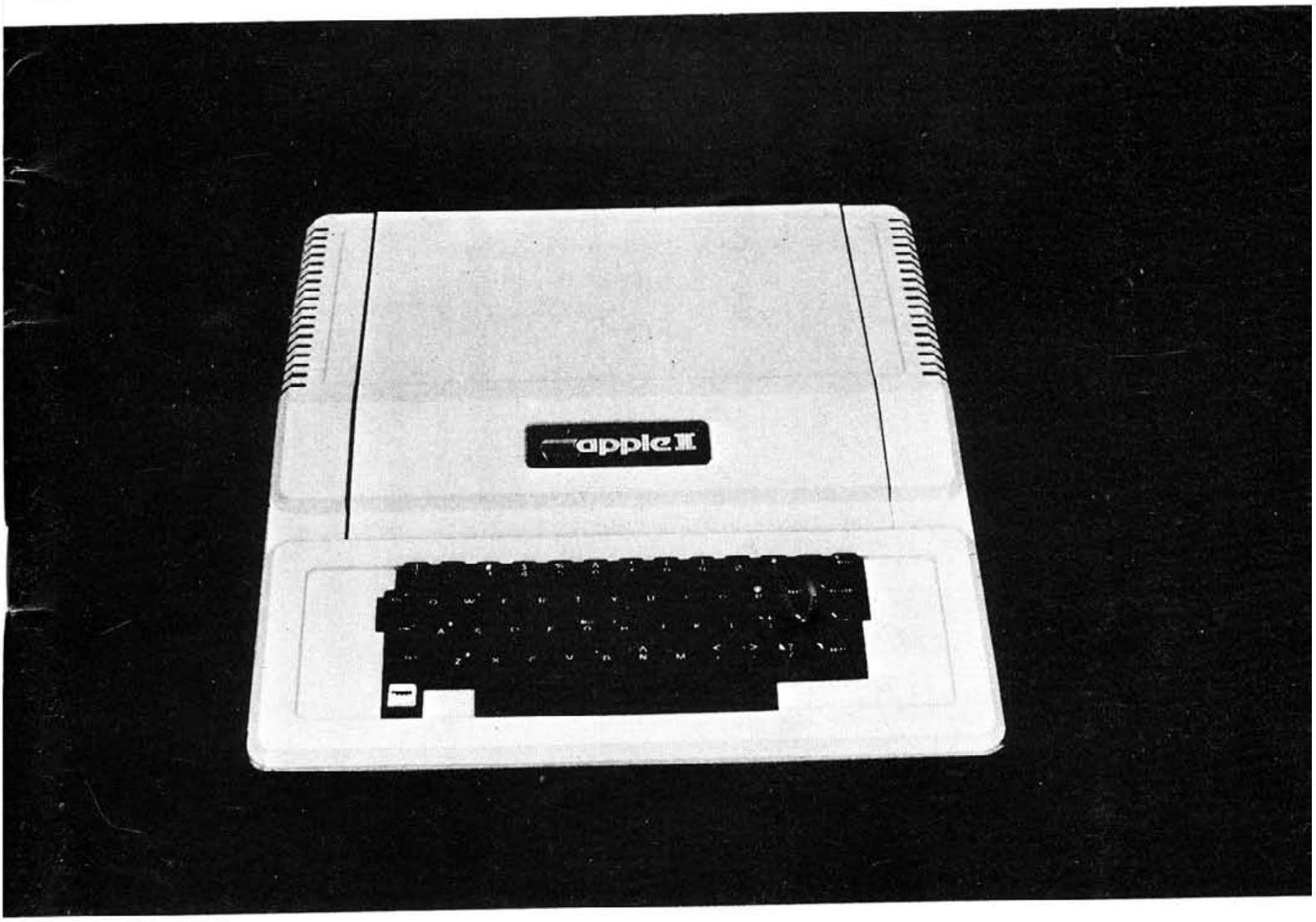
REPRINTED BY POPULAR DEMAND !

hard core
hard core
1&2

THE JOURNAL OF THE BRITISH APPLE SYSTEMS USER GROUP

JANUARY / MARCH 1981 £2 VOLUME 1 Nos 1 & 2



For everything **apple .. ring**

Lux Computer Services Ltd.

108 THE PARADE HIGH STREET WATFORD WD1 2AW
Telephone: Watford (0923) 29513

- Business Software • Professional Courses • Pascal •
- Recreational Software • Interfacing • Cromemco Systems •
- Level 1 Service Centre • Utilities • Colour Monitors •
- Supplies • Accessories • Enthusiasm • Open Tues - Sat •

Wida Software

2 Nicholas Gardens London W5 5HY

Computer Assisted Learning Educational Software Development

apfeldeutsch

Complete self-tuition package in German for Apple/ITT 2020: 9 diskettes, 6 audio cassettes, workbook, textbook. Takes Beginners to Post O'Level.
(Classroom version also available)
Special introductory offer until June 30:
Apple/ITT disks: £99

GERMAN FOREVER:

Stand-alone set of perpetual self-testing drills and routines in German - article and adjectival endings, word order, time of day, money, etc. Fine for O'level revision (BA Finals too!)
Apple/ITT disk: £20

TEACHER'S TOOLKIT:

Define and Write your own instant assessment tests - any subject - with no knowledge of computing.
Apple/ITT disk: £20

APPLEWRITER LOWER-CASE ADAPTOR

Beats Optrex! Give your eyes a treat and insist on lower-case for word processing (and foreign character sets) on screen with Applewriter. Plug-in ROM (Revision 7 Apples only)
£28

hardcore

THE JOURNAL OF
THE BRITISH APPLE SYSTEMS USER GROUP
P.O.BOX 174 WATFORD WD2 6NF

EDITED BY DAVID BOLTON

INTRODUCTION

The first issue of HARDCORE was in January of this year, and although we had less than hundred members we had 500 copies printed in the hope that the group would grow. Well, grow it did, and by the third issue we had a print-run of 2000.

With the membership now over 600, issues 1 & 2 are out of print, and by popular demand this combined reprint has been done.

In addition to the various authors and editorial staff of these first two Hardcores, we are indebted among others to Frances Teo and John Molloy for retyping and resetting the bulk of the contents.

David Bolton.

MEMBER OF THE
INTERNATIONAL APPLE CORE

Contents

EDITORIAL

Welcome, at last, to the first issue of the newsletter- and in a sense also the last, because we feel that the name 'newsletter' is not really suitable for the kind of publication that has developed.

In future, therefore, this publication will be referred to as the 'JOURNAL', and, we hope, will be issued bi- monthly.

I say 'we hope' because this depends on you, the members. The journal proper will be, in the main, a vehicle for the exchange of information between members (the dissemination of the bulk of information from other sources will be accomplished seperately- see below).

So I start my editorial on a subject that I expect to return to regularly- with a plea for contributions.

Don't worry if you consider that your writing skills aren't good enough- it's the idea that counts, and the editorial team will if required knock an article into shape for you (and then refer back to you for clearance before publication- so you needn't worry either about your ideas being distorted).

Talking to people at meetings, it appears that the other barrier may be that you don't consider yourself experienced enough to have anything worth saying. This is a fallacy.

Looking at the material going into this first issue, my own main criticism would be that there aren't enough articles directed towards beginners- who are the very people who need help and advice most.
The people who understand best the problems faced by beginners are .... beginners! So please let's have some articles- either answering problems or indeed posing them- this months question is, hopefully, an article for the next issue giving the answer.

The shortage of beginners articles in this issue is, unfortunately, not the only area under-represented and we hope in future to have a much more balanced magazine. So please send in readers letters, cartoons, software and book reviews, small ads, Adventure questions, etc, etc; as well of course as lots of articles.

THE STORY SO FAR

The British Apple Systems User Group was formed on 9th November 1980, when some forty Apple and ITT 2020 users met at The Old School, Branch Road, Park Street, St. Albans.

The name and aims of the group were discussed and decided upon, and a steering committee was elected, charged with the task of preparing a constitution to be put before a General Meeting of the group in the spring of 1981.

It was decided that meetings would be held on one Tuesday evening and one Sunday afternoon a month, and subjects covered so far have been the Alf Music Synthesiser Card and Speech Synthesis, as well as a 'Workshop Session'. Attendance has already grown considerably, and larger premises will probably have to be found in the very near future.

APFILE

I have mentioned above that the journal itself will primarily consist of original material contributed by members. This is because we will be exchanging newsletters with a large number of other user groups throughout the world, and to airmail to these

groups copies of our journal padded out with their own material doesn't make sense.

As a general rule, therefore, these newsletters will go into the club library with an index published in 'HARD CORE', and all the material of a general interest from these newsletters, as well as general interest items from the IAC's 'APNOTES' and other outside sources, will be published under the title 'APFILE', usually issued as a supplement to 'HARD CORE'. Having said this, there will of course be occasions where material from outside sources for various reasons appears in 'HARD CORE' itself.

MAIL

All mail should be addressed to the group's Post Office Box address,but to ease the problem of sorting please indicate on the envelope the subject of your letter.

PATIENCE

The group has only been established a very short time, and is growing fast. There is no paid administrative staff- all the work is done voluntarily. So if at first you have to wait a few days for a response- be it concerning membership, software library, Apnotes, or whatever- please be patient.

LOCAL GROUPS

One of the declared aims of the group is to set up local groups as soon as possible. In the meantime the committee are conscious of the problems of those members living too far from North London to attend meetings, and will ensure that you aren't treated as 'Second Class Citizens'.

If you would be willing to act as an 'Area Co-ordinator' please write in.

AND PRESSURE GROUP?

Another of the aims of setting up the group was to act as a 'pressure group' on manufacturers, distibutors, and dealers if necessary.

If any member has a problem where he feels we could be of help please say so. Conversly, if you feel someone deserves mention for particularly good service, use the journal as a platform.

AN APPLE BY ANY OTHER NAME

Back to the subject of international contact. The various mentions herein of something called an 'ITT 2020' may cause confusion to overseas readers.

The ITT 2020 is a version of the Apple II produced under licence by ITT. In general it is compatible from both the software and hardware viewpoints with the original Apple II, but uses a slightly different graphics system giving (usually non-fatal) problems with programs using Hi-Res graphics. It also has a different clock speed which causes problems with some hardware- notably the Microsoft Z80 card.

The other strange name that will appear from time to time is MICROSENSE- they are the sole distributors for Apple Computers Inc. in the United Kingdom.

DIRECT IMPORTS

Although the 'pound for a Dollar' situation of a few months ago has improved somewhat, U.K. prices for both hardware and software are still substantially higher than in the United States, even if taxes and transport costs are taken into account. While this situation continues there will be a strong

temptation for people to import equipment directly. Whether you buy a directly- imported Apple is of course your own decision, but you should be aware before doing so that conversion to UK standards is not just a matter of switching over the voltage- if you want to use a British T.V. rather than a monitor (and colour monitors are expensive!) then the crystal must be changed as well, which isn't a trivial task. You will also find that the established dealer repair network may be reluctant to provide service, so this is a point to clarify before you buy.

CONFESSION

Finally, a confession. I bought my ITT 2020 in November 1979- some fifteen months ago. I'm an accountant by trade, and had no computer background whatsoever. John Sharp has elsewhere in this issue confessed to a similar lack of background, and of the rest of the committee, only Martin Perry has any orthodox computer connection.

The point being, if you only bought your machine very recently and don't seem to be getting anywhere, don't despair! It will all drop into place- much more quickly, probably, than you can at present imagine.

DAVID BOLTON

CHAIRMAN'S CORNER

I didn't dream I would be in this position a year ago. I was having all sorts of problems with programming or there were faults with the machine; I didn't know which. I picked up various tips at exhibitions, but gradually learnt by experience. Then I overheard a conversation whilst looking through the magazines at Lion House, joined in and one thing led to another. In the next few weeks we had someone to ask, and could pass on some of our experience. Many telephone calls ensued and we had a number of meetings in the following months as well. We decided to set up a small contact group as there were others we were meeting as well.

Things moved rather slowly over the summer until the PCW show in September. I met a few people there including David Bolton. We discussed the idea of a user group when I came back off holiday and I began to realise I had let myself into something bigger than I had first planned. We discussed why Apple users seemed to be the last to set themselves up with a Users Group, and we were obviously in at what was probably the start of a national group. From then on it tended to take over our lives. Since then BASUG has really taken off.

The essence of the group is sharing. The most gratifying part of being part of such a group is just that, not being isolated, but having friends of a like persuasion. Obviously Apple owners are intelligent and discerning (they have gone for the best machine on the market for a start), but they also have lively minds and are creative as well. The content of this journal is evidence of all of this. If you haven't written anything yet,

now's your chance to participate. We want to know what you are doing with your Apple /ITT, and if we are making mistakes in not giving you what you want. It's not easy to please everyone but we can't please anyone if we don't know what they want. If you can't find anything to write about, remember, not everyone is advanced. Try to remember some of the early problems you had and pass them on to the newcomers. What about a review of that book you are struggling with, or thought was too easy or a waste of money.

BASUG has got off the ground, and it looks set to become one of the best User Groups- quality all the way from the machine and the members.

All the best in your Applecations in 1981.

JOHN SHARP

RELOCATING APPLESOFT

By Michael Mathison

Have you ever wanted to put your Applesoft program somewhere else in RAM other than from $800 onwards? (If you don't understand what this means then stop here because you won't understand the rest). Here's how:

Put the desired location plus one for your program to start at in $67,$68 in the usual reverse 6502 format, e.g. if you want to start at $4000 then type in the monitor

* 67 : 01 40 (and then return)

Then place a zero at the beginning of the new start, e.g. for $4000 type in the monitor

* 4000 : 00 (and then return)

Then type from the monitor CTRL-C followed by RETURN; type NEW followed by RETURN; and you are set up. Any program entered, LOADed or DOS LOADed will go in from $4000 (or wherever you wish) upwards.

This technique can be very useful when using high-resolution graphics on large machines, as an examination of the memory map in the Applesoft Reference Manual will show.

BUT BEWARE :

A DOS FP will clear this and put you back to $800 start.

A program SAVEd will not LOAD in the same place.

The NEW command DOES NOT CLEAR this setting.

PERSONALISED DOS COMMANDS

By Ian Trackman

When you give your Apple a disk command, DOS matches it up against a table of command words stored in memory locations $A884 to $A908 (decimal locations 43140 to 43272). (In this article, all references are to 48K computers. For 32K machines, subtract $4000 (decimal 16384) from all relevant numbers). You can print out the command table in a simple program :

```
10 LOCN = 43140
20 FOR I = 1 TO 28
30 PRINT "NO "I" AT "LOCN" IS ";
40 BYTE = PEEK (LOCN)
50 PRINT CHR$(BYTE);
60 LOCN = LOCN + 1
70 IF BYTE < 128 THEN 40
80 PRINT
90 NEXT
```

The byte holding the last letter of each command has its most significant bit set high, which we look for in line 70. This is how DOS knows when it has reached the end of each command word.

Provided that you keep the start of the command table at $A884 and you do not increase its overall length, you can change any or all of the command words to words (or secret codes !) of your own choosing. If you substitute shorter words, all of the following letters must be moved forward to fill the gaps.

For example, let's change the CATALOG command to DIR (short for "directory"). Here is a program to do it, which you might like to incorporate into a HELLO boot routine :-

```
10 BYTE = 43218
20 FOR I = BYTE TO 43272
30 POKE I, PEEK(I + 4)
40 NEXT
50 POKE BYTE, ASC ("D")
```

```
60 POKE BYTE + 1, ASC ("I")
70 POKE BYTE + 2, ASC ("R") + 128
```

In line 10, we set BYTE to the memory location of the first letter of the old command word, which we can obtain from the previous program. The number 43272 in line 20 is the last byte of the command table and remains constant.

As CATALOG has seven letters and DIR has only three, the whole of the subsequent part of the table must be moved forward by four bytes, which we do in lines 20 to 40.

Applesoft's ASC function leaves the most significant bit of the letter low, so that in line 70 we have to set the last byte of the new word high by adding 128 ($80).

If you alter the command table and then initialise a disk, your new commands will be incorporated into its DOS and will come into effect whenever you boot from that disk. It is beyond the scope of this article to explain how such changes can be written directly into an existing DOS on disk without re-initialising it, but for those who can handle RWTS, the DOS command table is on Track $1, Sectors $7 and $8 (DOS 3.3) and on Track $1, Sectors $A and $B (DOS 3.2).

HI-RES PLOTTING ON THE ITT 2020

By John Sharp

The ITT 2020 has a different resolution to the Apple. There are 192 points on the Y axis (the vertical one) on both machines, but on the X axis the ITT has 360 points whereas the Apple has only 280. This is achieved by storing the points in a different way. At this stage it is sufficient to say that the Apple uses 8 bits to store a point, while the ITT uses 9. When an Apple generated picture (say the Hopalong Cassidy picture) is run on the ITT then every 9th line is missing ,because there is no data in the range of memory the ITT uses for these lines. The extra resolution is gained by fitting more lines on to the screen in these places, instead of at the end of the screen. With something not completely filling the screen, it is not always as obvious. However even the Applevision program has the appearance of the man dancing behind bars.

This is not the case if the ITT generates a Hi-Res picture from Basic because it then follows the ITT Palsoft Hi-Res routines. There is a difference however in the picture produced. It is squashed in at the sides.A circle looks oval. Try running the following program to see this:-

```
10 REM CIRCLE WITH EQUAL DIVISION OF
POINTS
20 HGR : HCOLOR = 7
30 PI = 3.14159
40 R = 90: REM RADIUS
50 FOR N = 1 TO 40 : REM 40 POINTS
60 ANGLE = 2 * PI * N / 40
70 X = R * SIN( ANGLE )
80 Y = R * COS( ANGLE )
90 HPLOT X + 179, Y + 95
100 REM THE 179 AND THE 95 MOVES THE
CENTRE OF THE CIRCLE TO THE CENTRE OF
THE SCREEN RATHER THAN THE TOP LEFT AND
SO BRINGS ALL POINTS ONTO THE SCREEN.
110 NEXT N
120 END
```

On an Apple this will give a good circle which can be used to test your TV/monitor. On the ITT the circle will be flattened in at the sides, or depending on your point of view stretched vertically.

How can you fix this discrepancy? Quite easily, by using a factor of 7/9 (the ratio of the APPLE to the ITT screen) on the Y coordinates. Alter line 80 to:-

Continued on page 10

HEX/DEC & DEC/HEX CONVERSION

BY T.TSE

Following Applecalls (Practical Computing, Sept 1980), try this version for size and speed of execution. The routine submitted by T.Tse of Microsolve Computer Services Limited is all in machine code, occupies less than 80 bytes, and uses the not so well known Applesoft & command. It also accesses some utilities already present in the Apple Monitor and Applesoft Interpreter, while leaving your Applesoft program in memory undisturbed.

CALL 768 ($0300) initialises the Ampersand Exit Vector which is located at $03F5.
To save this routine, type in one of the following:

BSAVE HD-DH,A$300,L$4F (Disk)

300.34F W (Cassette)

The examples at the top of the next column illustrate how to use the conversion routines. There then follows a listing of the routine using the Apple Dissassembler. On the facing page is the original annotated Assembler program.

```
]PR#1
]CALL 768      :REM INITIALISE & EXIT VECTOR

]& H 100       :REM DECIMAL TO HEXADECIMAL
0064
]& D 100       :REM HEXADECIMAL TO DECIMAL
256
]& D 64
100
]CALL-151

*300LL

0300-  A9 4C     LDA  #$4C
0302-  A2 10     LDX  #$10
0304-  A0 03     LDY  #$03
0306-  8D F5 03  STA  $03F5
0309-  8E F6 03  STX  $03F6
030C-  8C F7 03  STY  $03F7
030F-  60        RTS
0310-  AA        TAX
0311-  F0 38     BEQ  $034B
0313-  20 B1 00  JSR  $00B1
0316-  E0 44     CPX  #$44
0318-  F0 0F     BEQ  $0329
031A-  E0 48     CPX  #$48
031C-  D0 F2     BNE  $0310
031E-  20 67 DD  JSR  $DD67
0321-  20 52 E7  JSR  $E752
0324-  A6 50     LDX  $50
0326-  4C 41 F9  JMP  $F941
0329-  A2 00     LDX  #$00
032B-  86 9E     STX  $9E
032D-  86 9F     STX  $9F
032F-  C9 3A     CMP  #$3A
0331-  90 02     BCC  $0335
0333-  E9 07     SBC  #$07
0335-  0A        ASL
0336-  0A        ASL
0337-  0A        ASL
0338-  0A        ASL
0339-  A2 03     LDX  #$03
033B-  0A        ASL
033C-  26 9F     ROL  $9F
033E-  26 9E     ROL  $9E
0340-  CA        DEX
0341-  10 F8     BPL  $033B
0343-  20 B1 00  JSR  $00B1
0346-  D0 E7     BNE  $032F
0348-  4C 28 ED  JMP  $ED28
034B-  4C 03 E0  JMP  $E003
034E-  00        BRK
034F-  00        BRK
*
```

AMATEUR RADIO

Edited by Donald Maclean

Will members who are licensed amateurs or short-wave listeners find time to write briefly to P.O.Box 174, Watford, with their Name, Address, (Callsign) and details of the radio applications of their Apple / 2020 which they have effected or wish to achieve.
The first list of known UK AppleHams is:-

G3DNQ G3LUO G3VDS G3XSF G4BIO G4CKW G4DDX
G8FAT G8GFH G8KIO G8LDR G8OON G8RAN G8VDQ
GM4BOM

Chairman of the Radio Amateur 'Special Interest Group' is WB7TRQ, Jim Hassler, ewho runs a library of ("about 80") ham-orientated programs and conducts an "Apple Net" on 14.329Mhz at 0100 gmt Mondays - ok for East and West coasts of USA but not ideal for G people or propagation!

TSE COMPUTERS APPLE II ASSEMBLER PAGE 01

```
0010:                    ID=13
0020:
0030:                    APPLE II
0040:                    SUPER-CONVERSION
0050:                    WRITTEN BY T TSE
0060:
0070: 0300                    ORG   $0300
0080:
0090: 0300 A9 4C      SETUP   LDAIM $4C     !SETUP JUMP INSTRUCTION
0100: 0302 A2 10              LDXIM ENTRY   !FOR APPLESOFT & COMMAND
0110: 0304 A0 03              LDYIM ENTRY   /
0120: 0306 8D F5 03           STA   $03F5   !AT AMPERV EXIT VECTOR
0130: 0309 8E F6 03           STX   $03F6
0140: 030C 8C F7 03           STY   $03F7
0150: 030F 60                 RTS
0160:
0170:                    MAIN CODE ENTRY FROM APPLESOFT
0180:
0190: 0310 AA         ENTRY   TAX           !ON ENTRY CHAR FOLL & IS IN ACC
0200: 0311 F0 38              BEQ   EXIT    !EXITS IF NO 'D' OR 'H' FOUND
0210: 0313 20 B1 00           JSR   $00B1   !SET APPLESOFT CHRGET TO NEXT CHAR
0220: 0316 E0 44              CPXIM $44     !IS SELECT A 'D'?
0230: 0318 F0 0F              BEQ   HXDEC   !YES GOTO HEX-DEC ROUTINE
0240: 031A E0 48              CPXIM $48     !IS SELECT AN 'H'?
0250: 031C D0 F2              BNE   ENTRY   !NO SCAN SOME MORE
0260:
0270:                    DECIMAL TO HEXADECIMAL CONVERSION
0280:
0290: 031E 20 67 DD   DCHEX   JSR   $DD67   !EVAL EXPRESSION INTO FP ACCS
0300: 0321 20 52 E7           JSR   $E752   !CONVERT TO UNSIGNED INTEGER
0310: 0324 A6 50              LDXZ  $50     !ACC SET - GRAB X-REG
0320: 0326 4C 41 F9           JMP   $F941   !PRNTAX AS HEX DIGITS
0330:
0340:                    HEXADECIMAL TO DECIMAL CONVERSION
0350:
0360: 0329 A2 00      HXDEC   LDXIM $00
0370: 032B 86 9E              STXZ  $9E
0380: 032D 86 9F              STXZ  $9F     !CLEARS 16-BIT INTEGER
0390: 032F C9 3A      LOOP0   CMPIM $3A     !IS DIGIT > 9 ?
0400: 0331 90 02              BCC   LOOP1   !NO - SO BRANCH
0410: 0333 E9 07              SBCIM $07     !YES- ADJUST·HEX
0420: 0335 0A         LOOP1   ASLA
0430: 0336 0A                 ASLA          !FOUR ARITH SHIFT LEFT
0440: 0337 0A                 ASLA          !GETS RH-NIBBLE INTO LH
0450: 0338 0A                 ASLA
0460: 0339 A2 03              LDXIM $03     !SET FOR SHIFT INDEX
0470: 033B 0A         LOOP2   ASLA          !GRAB CARRY A NIBBLE BIT
0480: 033C 26 9F              ROLZ  $9F     !ROL IT INTO INT NUMBER
0490: 033E 26 9E              ROLZ  $9E     !ROL SAME INTO HIGH BYTE
0500: 0340 CA                 DEX           !ASL COUNTER CHECK
0510: 0341 10 F8              BPL   LOOP2   !LOOP TILL FINISHED
0520: 0343 20 B1 00           JSR   $00B1   !GRAB NEXT CHAR BY CHRGET
0530: 0346 D0 E7              BNE   LOOP0   !LOOP TILL ENDLINE
0540: 0348 4C 28 ED           JMP   $ED28   !PRINT UNS INT AS DECIMAL
0550:
0560: 034B 4C 03 E0   EXIT    JMP   $E003   !RE-ENTERS CURRENT BASIC
ID=
```

Continued from page 7

```
80 Y = R * COS ( ANGLE ) * 7/9
```

Now rerun the program. If you are on an Apple you will have a circle squashed top and bottom into an ellipse. If you are on an ITT then you will have a perfect circle. If not your TV/monitor needs adjusting, or you have mistyped the program.

This problem affects all equations you wish to plot or the correct scale of anything you wish to plot on the ITT. To overcome it you need to scale all the Y coordinates by the factor 7/9, so that the plot is not distorted. This means that there are points nominally off the screen which can be brought on. If you are working with the circle program above , what is the maximum value for the radius R. On the Apple you would say 96 (192 /2). In fact it would have to be 95 since 95+ 96 (the centering addition) = 191 the maximum value of the Y axis coordinate. However, on the ITT you could have a radius of 122 (= 95 * 9/7) since the factor 7/9 would bring it back to 95 in line 80 and it would not go off screen. It is important to remember this when you are setting a window on the ITT screen in a program and also using the factor 7/9, otherwise the maximum screen size would not be used.

For shape tables there are problems associated with this 8-9 bit change. If you wish to preserve shape during rotation,

you might wish you had bought an Apple instead of an ITT. Depending on how the shape table generated a square, it can go into a diamond from a square or vice versa. It could be overcome if there was a way to alter the size of the X and Y directions independantly when scaling a shape from a shape table, but this is not possible directly from Basic. Is it possible to use a machine code routine that could be Poked in? If so write it up for a future column.

ADVENTURER'S FRIEND

By Keith Jones

Being a somewhat unhelpful guide, giving the answers to the commonest Adventure questions.

(Editors note: As this is the first issue of HARD CORE, Keith has put together some of the most frequently asked questions. For future issues, he will attempt to answer your specific questions)

MICROSOFT ADVENTURE

Q. I can't get past the snake.
A. A little bird can tell you how.

Q. The dwarf keeps killing me!
A. Not if YOU have the axe.

Q. How do I enter the tiniest place?
A. The answer is writ in letters of fire!!

(N.B. 'Y2' is part of the adventure, and not a program error)

ADVENTURELAND

Q. How do I wake the dragon?
A. I wouldn't bother.....

Q. How do I get the Blue Ox?
A. The same way you would catch a fish.

Q. How do I go below ground?
A. This one "stumps" me.

STRANGE ODYSSEY

Q. Is there something in this cave?
A. You "fire" me with enthusiasm!

Q. I can't figure out these controls.
A. A gentle touch works wonders!

MYSTERY FUN HOUSE

Q. How do I enter?
A. The price of admission "sticks" to you.

Q. Why do I keep getting thrown out?
A. Must be something about your appearance!

THE COUNT

Q. How do I get off the ground floor?
A. Try being "DUMB"

Q. How do I get through the locked door?
A. Try reading your mail!

VOODOO CASTLE

Q. I'm stuck in the cellar.
A. The solution is in the future!

Q. How do I keep from getting killed?
A. Try protecting yourself!

PIRATE ADVENTURE

Q. Help! I'm stuck on a window ledge.
A. You need a magic word.

PYRAMID OF DOOM

Q. How do I get into the pyramid?
A. Some keys would help.

MISSION IMPOSSIBLE

Q. How do I open doors?
A. You need proof of identity.

Q. They still won't open!
A. Sit and think about it!

I hope the above helps fellow- adventurers ... but not TOO much!!! My own series of Adventures is now almost complete and will be launched on the unsuspecting public very soon. Here's a resume of some of the most commonly used words in Adventures:

GET, PUT, TAKE, NORTH, SOUTH, EAST, WEST, N, S, E, W, UP, DOWN, INVENTORY, MOVE, SLIDE, LIGHT, FIRE, GO, PUSH, PULL, TIE, UNTIE, GRAB, DROP, USE, UNDO, TALK, SAY, RUB, DIG.

Lastly, a very sincere vote of thanks to SCOTT ADAMS, GREG HASSETT, etc, and to whoever wrote the very first Adventure for giving us many hours of pleasure (and frustration!!!). Good luck to all of you and please leave the caves as you would wish to find them!

EAMON ADVENTURE SERIES

We are fortunate in having access to this series,written by Donald Brown of the Colorado 'Apple Pi' club.

These adventures are outstanding in their concept of allowing a character (i.e you!) to develop from adventure to adventure by retaining skills and wealth already acquired in earlier games in the series.

The first disk in the series, comprising the Master Program and 'Beginners Cave', as well as utility programs to ENABLE YOU TO DEVELOP YOUR OWN ADVENTURES is available at club meetings or by post, to members only, at a price of FOUR POUNDS to cover media and distribution costs.

INTRODUCTORY DISK

The following is a summary of the programs on the introductory disk. We hope you find there is a good selection, and that we have ironed out the bugs.
If you do have problems, then before you contact anyone please check you have read the instructions. Also please note that the Keynote and Edit+ routines, because they interrupt the keyboard input routines, can interfere with some other programs. So it may be necessary to reboot DOS or clear memory some other way before running them.
If you like the selection, write to the newsletter. If you don't, write and criticise (but constructively please), especially if there are ways to make them run better.

1.INTEGER. Allows you to run the integer basic programs. You will have to BRUN it and then RUN the integer programs once you have the integer prompt > on the screen. You can get back to Applesoft with an FP command.

2.LIFE is a low res graphics version of John Horton Conway's famous simulation game, from the "Best of Micro Book 1". If you are not familiar with LIFE then see the seperate article in this journal.
Contributed by John Sharp.

3.HAUNTED CAVE. This is a slightly adapted version of "Spelunker", published in "Micro" magazine, OCTOBER 1979. It serves as a simple but reasonably challenging introduction to Adventure games; and being written in Basic (Integer), one can cheat by LISTing to find out valid commands, etc. To minimise the speed limitations inherent in a Basic program, you must use the command LOOK each time a description of your current location is required. The main

change from the program as originally published is that you now get more than one life (defaults to nine but can be user-defined) which makes the game less frustrating for beginners. A map is available for those who become hopelessly lost!
Contributed by David Bolton.

4.CHICKEN is an example of the use of shape tables to animate a chicken walking across the screen.From an article in "Creative Computing".
Contributed by Martin Perry.

5.APPLE ROSE is a hi-res pattern curve generator which, if left on its own, will continue to draw new patterns under random input. From "Recreational Computing".
Contributed by Warren Avery.

6.ZOMBIE ISLAND. The original version, for the PET, was written by Trevor Lusty and published in "Practical Computing". This game seems to be very popular, but the graphics, sound effects, and algorithms could all be drastically improved to take full advantage of the Apple's capabilities. Any offers?
Contributed by David Bolton.

7.FIVE GUESSES. A simple program originally published under the title "Word" in "Basic Computer Games".
Contributed by David Bolton.

8.HIRES DEMO is a set of programs which show some of the possibilities of using the hi-res screens.
Contributed by John Sharp.

9.HANGMAN. As submitted the vocabulary consists of 200 words, listed as DATA statements at five words to a line on lines 1005 to 1200 of the program. Line 1000 defines the number of words to be selected from, and this can be set to less than the available words to give an element of 'handicapping', but will crash the program if set higher than the available number of words.
Contributed by David Bolton.

10.BLACKBOX. This game is from "More Basic Computer Games", suitably amended to run on Apple Systems machines.
Contributed by David Bolton.

11.APPLE INVADER is a SPACE INVADER game from "Softside" magazine. It is slightly different to the usual Invader games, but great fun to play.
Contributed by Martin Perry.

12.NIGHTMARE is an Applesoft version of the game in the Apple Contributed Programs. You shouldn't look at the listing, just try and play.
Contributed by John Sharp.

13.GRANDAPPLE CLOCK is a hires simulation of an analogue clock, from "Creative Computing".
Contributed By Martin Perry.

14.SPACE MAZE is a very difficult game of skill, using the paddles to move along a maze. From "Nibble" No.1.
Contributed by Warren Avery.

15.KEYNOTE allows audible feedback from your keyboard. Each key returns a note through the internal speaker. The program comes from "The Best Of The Cider Press", and has been adapted for use with DOS.
Contributed by John Sharp.

16.EDIT+ is a program mainly for ITT users to give all the editing facilities of the Autostart Rom Machines, plus some more. From the "Best Of the Cider Press".Contributed By John Sharp.

17.MACHINE TO POKES ROUTINE is the program from the DOS manual with some instructions, since it isn't too clear what to expect from their write up.
Contributed by John Sharp.

18.BINARY FILE COPIER is a program from "Practical Computing", which makes it easy to transfer binary files from one disk to another or to tape.
Contributed by John Sharp.

19-22 SHAPE DESIGNER ,SHAPE-DES ASSEMBLER, SHAPER, AND SHAPE-PROG are a set of programs from "Apple Orchard", which allow you to make shapes easily into a vector table and then to assemble them into a shape table for your own program. If you have trouble with these programs, then it may be that you have some interference from other programs, particularly machine code ones. Reboot DOS and try again.
Contributed by John Sharp.

23.SINGLE DISK COPY allows you to copy a complete disk with only one drive. Just follow the instructions.

24.AUTOAPPLESOFT is a program from "Call-Apple" which adds the AUTO command in Integer Basic to an Applesoft program.
Contributed by John Sharp.

25.MINIASSEMBLER is the Miniassembler from the Integer listing, which has been relocated at $1000. It is poked in from Basic and can be saved as a binary file after you have run the program, if you so wish.
Contributed by John Sharp.

26.ORGAN. Contributed by David Bolton.

27.SEARCH/CHANGE is a program to append to your program, which allows you to search for variables or strings, and then change them. It originates from "Micro" magazine.
Contributed by John Sharp.

INTRODUCTORY TAPES

The programs on the introductory tapes are the ones that will run without requiring a disk. When you have a disk drive you can bring an empty disk along and copy the disk programs.

TAPE 1	TAPE 2
1. Chicken	6. Black Box
2. Apple Rose	7. Apple Invader
3. Zombie Island	8. Nightmare
4. Five Guesses	9. Grandapple Clock
5. Hi-res Demo	10. Space Maze

TAPE 3	TAPE 4 (INTEGER PROGRAMS)
11. Autoapplesoft	
12. Search/Change	16. Organ
13. Edit+	17. Haunted Cave
14. Keynote	18. Life
15. Mini-assembler	

If you find you have trouble loading the tapes, bring your own tape recorder to a club meeting and load them directly from disk onto your tape recorder, with the settings you normally use. There are programs to automatically load all the programs on each side of the tape. The club wishes to thank Michael Mathison for writing the routines to do this.

AIMS OF THE GROUP

By Martin Perry

What is a club? The Oxford Dictionary defines it as an "Association of persons united by some common interest, meeting periodically for co-operation".

Our club, "BASUG", is certainly that. We all have an interest in the APPLE or ITT2020 computer and our meetings are arranged with mutual co-operation in mind.

By capitalising on our members' different areas of expertise we can all learn more about the Apple system; and greater knowledge leads to greater enjoyment and fulfillment.

Obviously the group will be expanding rapidly and one of the aims of the group is the formation of special interest sub- sections which will concentrate on areas of specific interest. Some possibilities are sub- sections dealing with PASCAL, graphics, machine language programming, business and scientific applications.

Regular talks and demonstrations by Apple dealers and members will be an important part of our program. Topics already planned for future meetings include talks on Pascal, the graphics tablet, and a demonstration of low-cost printers.

The program library will be an important part of the group's function and should rapidly develop into a really useful asset. We plan to contact other user groups, particularly in the U.S and Canada, to exchange software and newsletters. This should lead to a source of new and original information and programs for the group.

The group's newsletter "HARD CORE" will be published as regularly as your contributions permit. Remember, it's your group and the newsletter is an ideal medium for exchanging ideas, problems, asking for help, or simply letting off steam. Short programs will be particularly welcome for a section called "Programming Quickies".

"Hard Core" will be sent on a reciprocal basis to as many other clubs as will co-operate. You can expect to see articles and contributions from many sources.

A book and magazine Library will be organised shortly and files of "APPLE" related articles will be kept. We are looking at a Database to keep track of these, and when we get the system in operation regular indices will be published in "Hard Core".

Finally, the most important aspect of our group will be the personal contact between members, anddiscussion sessions will be regularly scheduled in our programme to allow members to share views and experiences as well as giving us a chance to meet new members.

INPUT & PRINT USING

BY GEOFFREY CLEMENTS

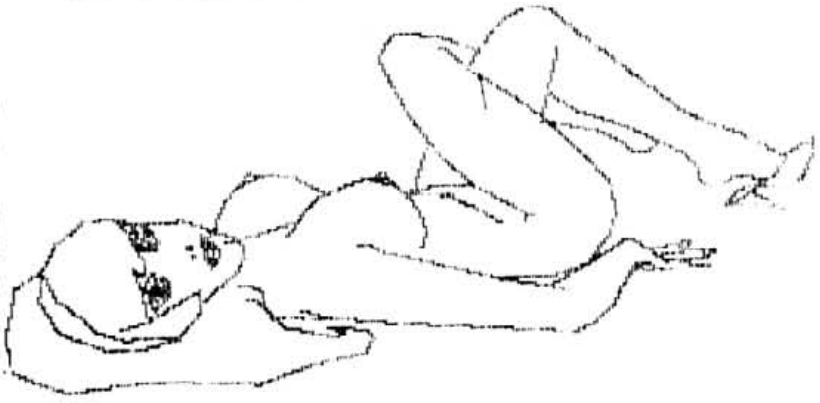
Here are two sub-routines to include in your programs to control the format of input and output respectively.

The input routine I find particularly useful. It prevents the user from typing any non-numeric keys when a numeric input is requested. It also saves the main program having to check on maximum length, range of values, etc., and prevents the need for an error routine to trap CTRL-C - the GET RI$ just rejects CTRL keys.
It is also possible to insert (for example at line 1025) a command such as:

```
1025 IF RI$ = CHR$(27) THEN 9000
```

so that if the ESCape key is pressed in response to an input request, the program will branch to (for example) an exit routine.

The PRINT USING routine is unfortunately quite slow, and I would appreciate any suggestions on improving the coding. The ideal solution would be, of course, to use a machine code routine.



```
1 B$ =  CHR$ (7): REM  SET B$ IN
    PROGRAM TO EQUAL WARNING WHE
    N INVALID KEY PRESSED (I.E.
    BELL CHARACTER OR NOTHING)
100  TEXT : HOME : VTAB 5: PRINT
    "PLEASE ENTER A NUMBER ?"
110 V = 5:H = 30:M = 3:M1 = 1: GOSUB
    1000
120 N =  VAL (R$): REM  N WILL BE
     BETWEEN 0 AND 999
130  VTAB 7: PRINT "PLEASE ENTER
    A STRING ?"
140 V = 7:H = 30:M = 5:M1 = 2: GOSUB
    1000
150  REM  R$=STRING RETURNED
160  STOP
1000  VTAB V: HTAB H:R$ = ""
1010  GET R1$: IF R1$ =  CHR$ (13
    ) THEN 1080
1020  IF R1$ =  CHR$ (8) THEN  IF
     LEN (R$) <  > 0 THEN R$ =  MID$
    (R$,1, LEN (R$) - 1): PRINT
    R1$;" ";R1$;: GOTO 1010
1030  IF  LEN (R$) + 1 > M THEN  PRINT
    B$;: GOTO 1010
1040  IF M1 = 1 THEN  IF (R1$ < "
    0" OR R1$ > "9"), AND R1$ <  >
    "." THEN  PRINT B$;: GOTO 10
    10
1050  IF M1 = 2 THEN  IF R1$ < "
    " OR R1$ > "Z" THEN  PRINT B
    $;: GOTO 1010
1060 R$ = R$ + R1$
1070  PRINT R1$;: GOTO 1010
1080  PRINT : RETURN
1090  REM  ************************
    *
1100  REM  R$ = RETURNED STRING
1110  REM  SET M TO MAX LENGTH
1120  REM  SET V,H TO SCREEN POSN

1130  REM  SET M1 TO 1 FOR NUMERI
    C
1140  REM              2 FOR ALPHA
1150  REM  ************************
    *
2000  REM  ************************
2010  REM  G.I.CLEMENTS MAY 1980
2020  REM  SECOND CITY SOFTWARE
2030  REM  24 PEBBLE MILL ROAD
2040  REM   BIRMINGHAM B5 7SA
2050  REM     021-472-0703

2060  REM  ************************
```

```
1  REM  LINES 100-120 ARE JUST A
    SHORT EXAMPLE OF HOW THIS
2  REM  ROUTINE CAN BE USED IN A
    PROGRAM
100  FOR I = 1 TO 1.5 STEP .0098
110 X = I:S = 10:D = 3: GOSUB 100
    00
120  PRINT X$: NEXT
130  PRINT : STOP
10000 X =  INT (X * 10 ^ D + .5) /
     INT (10 ^ D + .5):X$ =  STR$
    (X): FOR II = 1 TO  LEN (X$)
    : IF  MID$ (X$,II,1) <  > ".
    " THEN  NEXT :X$ = X$ + "."
10010  IF II = 1 THEN X$ = "0" +
    X$:II = 2
10020 X0$ =  MID$ (X$,II + 1,D)
10030  IF  LEN (X0$) <  > D THEN
    X0$ = X0$ + "0": GOTO 10030
10040 X$ =  LEFT$ (X$,II) + X0$
10050  IF  LEN (X$) < S THEN X$ =
    " " + X$: GOTO 10050
10060  RETURN
10070  REM  D=NO. OF D.P.'S
10080  REM  S=NO. OF SPACES
10090  REM  X=NUMBER TO FORMAT
10100  REM  X$=STRING RETURNED
10110  REM  ************************

10120  REM  G.I.CLEMENTS MAY 1980

10130  REM  SECOND CITY SOFTWARE
10140  REM  24 PEBBLE MILL ROAD
10150  REM   BIRMINGHAM B5 7SA
10160  REM     021-472-0703

10170  REM  ************************
```

(Editors Note: Both these routines are in the Software Library. The '& PRINT USING' program recently published in 'NIBBLE' answers Mr Clements' request for a fast machine-code utility,and this also will shortly be available from the Library. The disadvantage of a machine language routine, particularly one as comprehensive as &PRINT USING, is that it makes a program using it less portable- for cassette based systems, for example, the mechanical problems of storing and loading a machine-language sub-routine are formidable. Unless speed is of over-riding importance the use of simple basic sub-routines contained within the program has a lot of advantages.)

INTERNATIONAL APPLE CORE

BASUG is now a Full Member of the International Apple Core.

The IAC is a non profit organisation composed of Apple computer user groups throughout the world. The IAC was formed to disseminate all types of information about the Apple from Apple clubs ,users manufacturers and Apple themselves. Although Apple were the initiators, apart from their being one of the sponsoring members, the IAC is independent of the Apple Organisation.

What does the IAC offer us as members ?

i) It publishes "The Apple Orchard" which, if the first two issues are a measure, promises to be a very good magazine. When the shipping problems are sorted out we shall have cheap copies (after the initial free one we promised).
ii) There will be a regular distribution of Technical notes, APNOTES, consisting of software and hardware. A summary of the set we have received to date, plus details of how to get them is outlined in a separate article.
iii) A regular disk of SOFTWARE will come over about once a month. This will go into the library,initially, and distribution details will be published in the journal.

There are also special interest groups (SIG's) covering Education, the Handicapped, Legal, Ham Radio and Medical aspects of the Apple.

The other important reason for a union of User Groups is contact between these groups . The way we shall have most contact initially is by participating in the newsletter exchange.

We have already received a large amount of information on the IAC in the way of minutes of meetings, a news bulletin, SIG contacts etc. You will probably want to read these but not necessarily keep copies of all of the information. Duplicates will be put into the library, and arrangements will be made to either buy photocopies at a nominal charge or loan copies by post. It would be appreciated if those members with photocopying facilities could put at least one extra copy back when they return the copy . By having such a method, enough copies can be built up to cut down the delay in spreading the information to everyone. PLEASE BE PATIENT, it will take a few weeks to get organised. There is a backlog for us to cope with since the IAC has been generating information for 18 months and we have received it all in one go.

Having seen that we will benefit in many ways from the IAC, it is not only fair but necessary that we put something back. Like our own Group, they rely on volunteers and can only survive if members are willing to give as well as take. It will also help us to gain prestige if we are a regular contributor. One way we will feed information in is by articles in the newsletter getting into the APNOTES and the other is by programs for their software disk. This not only gets BASUG a name but you will have a personal audience of tens of thousands.

The information available is as follows:
1. Minutes of IAC meetings to mid 1980- about 33 pages including maps of regions.
2. Summaries of types of membership- 3 pages.

3. SIG contact addresses, Director and Officer addresses, software and newsletter exchange information, SOURCE communication information- 5 pages.

4. December News Bulletin- 2 pages

5. "Apple Orchard" guidelines for authors- 5 pages .

HI-RES PAGE SWITCHING

By Michael Mathison

Any Apple or ITT 2020 with 24K or more RAM has two pages of high res. graphics available. (If you have DOS then 36K or more). To most BASIC programmers this seems unnecessary, only providing a means of using high res graphics with large programs, (using HGR2 instead of HGR). However, with a few simple POKES many useful effects such as animation can be achieved.

When you execute any high res. command (HPLOT,DRAW etc.), Applesoft and Palsoft look in location 230(decimal) i.e. $eE6 (hexadecimal) to tell it which page of high res graphics to plot on :

If 230 contains 32($20 in hex) it plots on page 1,
If 230 contains 64($40 in hex) it plots on page 2.

This occurs IRRESPECTIVE OF WHICH PAGE IS BEING DISPLAYED ,so that you can plot on page 2 whilst displaying page 1, and vice versa.

Try this on your machine :

```
HGR2:HGR
```

This clears both pages, and leaves you on page 1. Then :

```
HCOLOR = 3
HPLOT 0,0 TO 274,159
```

No surprises- you get a white diagonal line. Now:

```
POKE 230,64
```

This sets all plotting on page 2.Now type:

```
HPLOT 279,0 TO 0,191
```

Nothing happens! That is because the line has been drawn on page 2, not page 1. Now to look at page 2:

```
POKE -16299,0
```

This as you may know switches all display from page 1 to page 2 (see pages 132 to 133 of the Applesoft reference manual).

You will see the line you just plotted,and a nasty mess where the text should be. This mess is page 2 of text information, which is very hard to use; that is why HGR2 displays full screen graphics.

Now we are ready to try some simple animation:

```
10 HGR:HGR2:REM CLEAR BOTH SCREENS
20 REM WE ARE NOW ON PAGE 2 FULL
SCREEN
30   A=2
40   POKE 230,32:REM PLOT ON PAGE 1
50   POKE -16299,0 :REM DISPLAY PAGE 2
60   A=A-2:HCOLOR =0:GOSUB 1000
70   REM ERASE OLD SQUARE
80   A=A+2 : HCOLOR = 3 :GOSUB 1000
90   REM DRAW A NEW SQUARE
100  A= A+ 1
110  POKE 230,64:REM PLOT ON PAGE 2
120  POKE -16300,0 :REM DISPLAY ON PAGE 1
130  A = A-2 : HCOLOR =0: GOSUB 1000
140  A = A+2 : HCOLOR =3: GOSUB 1000
150  A = A+1
160  IF A < 80 THEN 40
170  END
1000 HPLOT A,A TO 100+A,A TO 100 + A,100+A
TO A,100+A TO A,A
1010 RETURN
```

For an ITT 2020 make line 1000 :

```
1000 HPLOT A,A to 100+A,A :HPLOT TO 100 +
A,100+A:HPLOT TO A,100+A:HPLOT TO A,A
```

If you read it carefully, and understand it you should find this technique very useful. This is a very simple example and gives no indication of the full scope of the technique. If you remove lines 40,50, 110 and 120 and then change 160 to IF A< 80 THEN 60, you can see the process without the switching.

APPLESOFT PAGE ZERO MEMORY MAP

HEXLOC	DECLOC	NAME	DESCRIPTION
00-02	00- 02		Warm start vector
03-05	03- 05		Vector to print string
0A-0C	10- 12		USR function Jump instruction
0D	13	CHARAC	Search Character) Used by
0E	14	ENDCHR	Scan-between-quotes Flag) STRLT 2
0F	15		Input Buffer Pointer : # of subscripts
10	16		Default DIM flag
11	17	VALTYP	Flags last FAC type: 0=Numeric,FF=String
12	18		Type: 80=Integer,00=Floating Point
13	19		Flag: DATA scan, LIST quote, memory
14	20	SUBFLG	1.Subscript Flag: 0=Subscripts allwd, 80=No Subscripts 2.FNX flag
15	21		0=INPUT, 40=GET, 98=READ
16	22		Comparison Evaluation Flag
17	23		Counter/Flag - Purpose?
20-4F	32- 79		Monitor Reserved Locations
50-51	80- 81	LINNUM	Gen Purpose 16-bit (Integer) number- Used by GOTO, etc
52	82	TEMPPT	Last Used Temp String Descriptor) Pointers for
53	83	LASTPT	Last Used Temp String Pointer) Descriptor Stack
54	84		)
55-5D	85- 93		Descriptor Stack (Temporary Strings)
5E-5F	94- 95	INDEX	Temp Pointer for moving Strings) Utility
60-61	96- 97		) Pointer area
62-66	98-102		Product Area for Multiplication/Division
67-68	103-104	TXTTAB	Pointer- Start of Program Text (Normally $0801)
69-6A	105-106	VARTAB	Pointer- Start of Simple Variable storage
6B-6C	107-108	ARYTAB	Pointer- Start of Array storage
6D-6E	109-110	STREND	Pointer- Top (i.e. End) of Array storage
6F-70	111-112	FRETOP	Pointer- Bottom of String storage (moves down)
71-72	113-114	FRESPC	Temp pointer-String storage routines(Utility String Ptr)
73-74	115-116	MEMSIZ	HIMEM- Highest location plus one available to Applesoft
75-76	117-118	CURLIN	Current Line No (FF= Direct Mode)
77-78	119-120	OLDLIN	Previous Line No (Set by CTRL=C, END, or STOP)
79-7A	121-122		Old Text Pointer- Loc for statement to be executed next (Used by CONT)
7B-7C	123-124	DATLIN	Current DATA Line number
7D-7E	125-126	DATPTR	Current DATA Address (Addr in memory from which data is being read)
7F-80	127-128		Input Vector- Set to $201 during INPUT and DATPTR ($7D/$7E) during READ
81-82	129-130		Current Variable Name
83-84	131-132	VARPNT	Current Variable Address (Used by PTRGET)
85-86	133-134	FORPNT	Variable pointer used by FOR/NEXT
87-8F	135-143		Work Area: Pointers, etc
90-92	144-146		Jump Vector for functions

HEXLOC	DECLOC	NAME	DESCRIPTION
93	147		?
94-95	148-149	HIGHDS	) Used by
96-97	150-151	HIGHTR	) BLTU
98-99	152-153		?
9B-9C	152-153	LOWTR	Gen Purpose Register- see BLTU, GETARYPT, FNDLIN
9D-A2	157-162		1. 9D-9F DSCTMP- Temp String Descriptor 2. 9D- Accum #1 Exponent, 9E-A1- Accum #1 Mantissa, A2- Accum #1 sign
A3	163		Series Evaluation Constant Pointer
A4	164		Accum #1 High-Order (Overflow)
A5-AA	165-170		Accum #2 Exponent/ Mantissa/ Sign
AB-AC	171-172	STRNG1	1. Pointer to a String- see MOVINS 2. AB-Sign Comparison Acc1&2, AC-Acc#1 lo-order(Rounding)
AD-AE	173-174	STRNG2	1. Pointer to a String- see STRLT2 / 2. Series Pointer
AF-B0	175-176	PRGEND	End of Program Text-Used by LOAD/SAVE but not RUN/LIST
B1-C8	177-200	CHRGET	CHRGET sub-routine- Entry here increments TXTPTR
B7	183	CHRGOT	Alt. Entry point- entry here doesn't incr. TXTPTR
B8-B9	184-185	TXTPTR	Pointer to next Character or Token
C9-CD	201-205		Random Number
D0-D5	208-213		Hi-res graphics scratch pointers
D6	214		Mode Flag
D7	215		?
D8	216	ERRFLG	$80 if active, 00 if inactive
D9	217		? (Part of ONERR)
DA-DB	218-219	ERRLIN	Line # where error ocurred
DC-DD	220-221	ERRPOS	TXTPTR save for HNDLERR
DE	222	ERRNUM	Type of Error
DF	223	ERRSTK	Stack Pointer value before error
E0-E2	224-226		Hi-res graphics co-ordinates XL, XH, Y.
E4	228		Hi-res graphics color byte
E5	229		?
E6	230	HIPAG	Hi-res page to plot on- $20= Page1, $40= Page2
E7	231		?(Shape routines)
E8-E9	232-233		Pointer to beginning of Shape Table
EA	234		Collision Counter for Hi-res graphics
F0	240	FIRST	Used by PLOTFNS
F1	241	SPDBYT	Speed = Delay#
F2	242		?
F3	243	ORMSK	Mask for flashing output
F4-F7	244-247		?ONERR pointers
F8	248	REMSTK	Stack Pointer saved before each statement
F9	249		?(Shape routines)

(This map isn't yet complete, and is published with a request for information on corrections and additions, so that a definitive map can be published in a later issue.)

BEGINNERS PAGE

By John Sharp

(Advanced programmers might learn something too)

There are a number of thick manuals with your Apple which you just cannot read and digest all at once . If you have a knowledge of Basic you can start programming straight away. However, you will still encounter the quirks of the Apple, as you would with any machine. Some of the items below you might come across in the manual, some you might learn by accident, whereas others you read about or someone tells you about them. The manuals are not always as clear as they could be so if you think you can write it up more clearly then do so.

POKE 33,33 ON THE SCREEN AND THE PRINTER

One of the first problems you will come across, especially if you are not a good typist, is the problem of editing a mistyped PRINT statement which has a string which is more than 40 characters long. You then put up a listing to alter a particular word, and having altered it run along with the cursor to copy the line. Then you run the line and lo and behold there is a long space in the middle, usually in the middle of a word. What has happened is you have copied this space in by adding to your string the space at the end of a line and the beginning of the next line that the Apple puts up to make listings easier to read. The screen is set to 40 characters but the listing does not use all of them. An example is easier to demonstrate than put down in writing, but let's try :-

```
10 PRINT "THIS IS A LINE OF RUBB
ISH TO EXPLAIN"
```

becomes on listing :-

```
10 PRINT "THIS IS A LINE OF RUBB
   ISH TO EXPLAIN"
```

If you copy this with the cursor and then run the line instead of:-

```
THIS IS A LINE OF RUBBISH TO EXPLAIN
```

you will get:-

```
THIS IS A LINE OF RUBB          ISH TO
EXPLAIN
```

There are a number of ways to overcome this problem. You could type the line again, which is a bit time consuming. You could put end of quotes, followed by a semi colon and reopen quotes, so that the printing carried on on the same line. You could follow the procedure of moving the cursor along with ESC A or ESC K so that it did not exactly copy but just moved it physically. This has some drawbacks in that you can easily add one space that you didn't intend to; it is also time consuming, especially if you have to use ESC A because you do not have the Autostart ROM.

The best way is to use POKE 33,33 before you list the line. However before you do this do a CALL -936 or HOME to clear the screen. (To see why this is necessary list a whole page then do a POKE 33,33 and press the RETURN key a few times.) Now list the line you wish to edit which will look like:-

```
10 PRINT "THIS IS A LOAD OF RUBB
ISH TO EXPLAIN"
```

Now run the cursor along and copy the line and you will see that there is no space at the end and beginning of the line ; it will copy the string faithfully. When you run it you will have no problems at all.

Before running however, type TEXT or POKE 33,40 to return you to the correct screen window width. It is a good idea to have the first line of any prorams as

```
TEXT:HOME
```

to not only clear the screen, but set the correct window.

If you have a PRINTER then POKE 33,33 can help you as well. Before you print a listing POKE 33,33, and instead of your getting a one sided 40 column listing then it will go right across the page which not only makes it easier to read but saves paper as well.

PUTTING MACHINE CODE PROGRAMS INTO YOUR APPLE

It is often hard enough to cope with Basic when you first start with your Apple, but you often want to use a machine code program as well. A listing in a magazine might use a machine code routine as a means of speeding up the program, or to put a shape table in for graphics. It is easy to do. You dont't have to understand it to use it, providing there are no bugs in it of course. The procedure is in the Red Book or the Reference Manual, but if you are not familiar

with machine code you may not know what to do or what to load from a particular listing.

The simplest listing is the machine code in hex. This will look something like:-0- A9 AC A2 10 A0 03 8E F6

In order to type this in, call -151 to get into monitor, which should give you the " * " prompt. Now type the number at the beginning of the line (i.e. 300)and follow it with a semi-colon (" : ") so that you have:-

300:

NOT 300- , this is a listing start and not an input start.

Now type the numbers as they are in the rest of the line with a space between them, then press return.

Now if you continue with the next line in your program in the same way you can load the whole program into memory.

You might have a program like the Hex to Decimal converter in this issue of "Hard Core" which has been diassembled. If we look at the first few lines they are as follows:-

```
0300- A9 4C     LDA #$4C
0302- A2 10     LDX #$10
0304- A0 03     LDY #$03
0306- 8D F5 03  STA $03F5
```

If you compare with the above line 300 then the numbers in the second column taken in sequence will be seen to be the same. These are the ones you want to type in, in the way described above, and ignore the rest. When you have gone all the way down the page check that you have not made any errors by typing 300L and pressing return. The listing should come up exactly as it is printed in the magazine. If one line (and almost certainly some below) are wrong, type that line number followed by a semi-colon, as before, and then press Return. Do the 300L listing and recheck. If the part you can see is OK then just type L, followed by Return and you will see the rest.

Now you want to save it onto tape or disk. For TAPE then type

300.34FW

followed by Return. This is interpreted as:- write the memory locations 300 to 34F (in hex) onto your tape recorder. When you come to load it back into the machine then you will need to type

300.34FR

and press Return when the tone starts. This is interpreted as READ into the loacations 300 to 34F.

If you have disk then to load the program, go back into Basic by typing CTRL C, then type:-

BSAVE HEX - DECIMAL , A$300,L$4F

and press return.

BUSINESS DATA STORAGE AND RETRIEVAL

By Nik Spicer

99% of business programming involves simply storing and retrieving information. Whether the application is a straight-forward stock control or an encyclopedia- like reference file, the aim is always to find the maximum information from the minimum of clues in as fast a time possible.

Given infinite search speed the programmer would not have to worry about indexing the data; given infinite storage the programmer could index everything. We do not have either, and so data management programs are mostly to do with juggling various limitations and compromises.

The major limitation that occurs is simply that everything happens in memory. Anyone who has contemplated an alphabetic sort of 400 random access files will know what this means. Just as humans cannot count the frequency of a particular word in a book without closely scanning the book, a computer cannot find a particular item on a disc or in memory without looking at everything there.

In both cases an index can help, but only if the author/programmer and reader/user agree on what is important.

Take the example (as computer courses often do) of a simple stock control system. Ideally an index should not be required since the stock item's code COULD be the record number. But it never is.

So we can construct an index of the stock code / record number. However, computers should make life easier, not harder, and who wants to remember a stock code? Now we need an index of the item descriptions, and, to follow the idea through, an index of suppliers. Index everything and we run out of disc.

Somewhere a compromise must be made between speed and bulk information. The fastest search of information is in memory, not on disc, so one option is to load the entire index into RAM and search that. The load need only occur once, with a save before quitting if the index has been changed. However even 48K is not a bottomless pit, and micros have lousy garbage collection, meaning that your Apple system will soon force a FRE(0) on you, particularly if you sort the indexes in BASIC.

If you try this method, do not fall into the trap of duplicating the index in the record. The second trap is actually sorting the index at all.

One idea that appeals is to forget 200+ character records altogether. Put each field into it's own random access record!

FIELD 1 CHRS FIELD 2 CHRS

ITEM CODE 10 DESCRIPTION 30

Then simply Open a file called "ITEMS,L10" and a file called "DESCRIPS,L30", etc.

Now every field is it's own index, and on finding the required field record, pull out the matching field record numbers and display. If you really must sort the fields then you will have to retain an array of record numbers to carry around with the fields.

This does not solve the problem of memory. If you are not holding a valid index in memory it is possible to avoid embarrassing FRE(0)'s by deciding at which point ALL data in memory is redundant, and CLEAR the memory - not forgetting to GOSUB through the DIM's, etc. Or (why not?) just RUN again.

The major objection to this system is simply that information in fixed length fields will either be too long or wasteful of disc space. The most frugal method of storage on disc is sequential filing, but this is the most wasteful in time and creates the most garbage in memory.

The following list represents the main methods of data storage and each one's merits or otherwise in time, memory use, disc space and storage capability.

Memory only e.g. 'PHONE LIST'

Very fast retrieval time with zero memory usage, since the data is held within the listing and therefore variable pointers point always within the program. The data stored is severely limited to the size of the program holding it. Disc storage is a matter of resaving the program at the end of each run.

Sequential storage only.

Slow retrieval mainly because indexing is pointless. Storage severely limited by memory size unless 'pseudo' random records are created using 'POSITION' command- realistically rarely viable since random files do it better. Memory choked by dead characters. Disc storage excellent in theory, but of no practical benefit.

Random Access only.

Slow retrieval, unless record numbers have obvious meaning. All comments given regarding sequential files apply except total size of file can exceed memory size, with forced FRE(0)'s further slowing searches if memory exceeded.

Indexed Random Access.

Greater disc space, increased programming problems, index file restricts memory but gains speed IF the index is relevant to the required search. Limited search capability, or more frequent FRE(0)'s.

Field length Random Access.

Wasteful of disc space, good search capability, high risk of FRE(0), complex programming, totally inflexible field length.

In addition, there are various proprietory systems which, by carefully matching the method to the application, can optimise performance within their design limits, for example:-

DDA System (available as D/DATABASE)

Restricted record sizes, highly complex programming, limited application, special data discs, often wasteful of disc space BUT entire disc available, very fast access (800 indexed files in 20 secs), zero memory usage (no garbage), complete search facilities, out-of-program-environment editing of files.

APPLESOFT "INPUT ANYTHING" ROUTINE

By Ian Trackman

Characters can be entered in response to a string INPUT command (e.g. INPUT A$) either as a string e.g. "THIS IS A STRING", with the first non-space character being quotation marks (the closing quotation marks are optional) or as a literal e.g. THIS IS A LITERAL, with no opening quotation marks. In the first case Applesoft drops the quotation marks and stores THIS IS A STRING in memory. The second type of response is stored as typed.

Most programs do not insist upon the first type of response, since it requires the user to remember always to type quotation marks as the first character of the input. However, it has the advantage that commas and colons can be entered without causing an EXTRA IGNORED error message and the consequent loss of the characters following the comma or colon. On the other hand, any characters (except blanks) following the second quotation marks will cause a REENTER message. The advantage of using the literal-type response is that one or more quotation marks will be accepted anywhere in the response except, of course, as the first character.

If this is not confusing enough already, the problem becomes even worse when loading text from disk. Data is loaded using the INPUT command but DOS can use only the literal-type response. The result is that if you have entered a string such as "A:B:C" and saved it to a text file on disk, when you try to reload it, INPUT will stop at the colon and reject :B:C with an EXTRA IGNORED message. Furthermore, when saving to disk, commas are treated in the same way as semi-colons are used in PRINT statements for supressing carriage returns. All of these different limitations make the use of commas, colons and quotation marks in string inputs a disaster-prone exercise.

To solve the problem, I describe below a machine-code subroutine which can easily be incorporated into an Applesoft program. The routine uses the Monitor's own input routine, which does not suffer from any of the restrictions mentioned above. Table 1 lists the subroutine's assembly language source code.

To use the subroutine, initialise a dummy string variable (call it X$ or IN$ or whatever you like) as the very first variable - of any type - in your program e.g. X$ = "" or IN$ = "X". It does not matter what appears between the quotation marks.

Your Applesoft program should POKE the machine-code subroutine into any memory space where you have 23 free bytes (memory locations 768 to 790 ($300 to $316) are free unless specifically otherwise used by your program). The decimal equivalents of the hexadecimal machine-code are set out in the DATA statement in the sample program below.

You can still use INPUT in your program in the normal way but if you want to allow the user to enter commas, colons and quotation marks, make your program CALL the subroutine at that point instead. Then re-assign the response from the dummy string variable to the string variable where you actually want to store the response, using the MID$ function. For example :-

```
10 IN$ = "" : REM THIS IS THE DUMMY STRING
VARIABLE
20 FOR I = 768 to 790
30 READ X
40 POKE I, X
50 NEXT
60 DATA 162, 0, 32, 117, 253, 160, 2, 138, 145,
105, 200, 169, 0, 145, 105, 200, 169, 2, 145, 105,
76, 57, 213 : REM FOR APPLESOFT IN RAM, CHANGE THE
LAST NUMBER (213) TO 13
.
.
1000 PRINT "WHAT IS YOUR ADDRESS ? ";
1010 CALL 768
1020 AD$(4) = MID$(IN$,1) : REM THE RE-ASSIGNMENT
```

The reason for using the unusual MID$(IN$,1) command is to force Applesoft to copy the string from the keyboard buffer at $200 and then save it as a new string in the string storage area at the top of memory so that it will not be destroyed by the next keyboard input.

The subroutine handles back and forward spacing in the same way as Applesoft does. The only difference is that it does not recognise Control C as a means of interrupting your program at an INPUT command.

To use the subroutine in disk loading routines, where you would normally have the command INPUT A$, type instead CALL 768 : A$ = MID$(IN$,1) with A$ as the required storage destination.

TABLE 1

```
; APPLESOFT "ACCEPT ANYTHING" INPUT ROUTINE
    ORG   =  $300    ;But can be
                     anywhere
                     convenient.
300 A2 00    LDX #0      ;Clear X Reg.
                     for NXTCHAR's
                     use.
302 20 75 FD JSR  $FD75  ;Monitor INPUT
                     routine
                     (NXTCHAR).
305 A0 02    LDY #2      ;X now holds
                     length of
                     input.
307 8A       TXA         ;Your string
                     is entered by
                     NXTCHAR
                     at $200,
308 91 69    STA  ($69),Y ;so tell
                     Applesoft
                     where it is &
                     its length.
30A C8       INY         ;$69,$6Apoint
                     to start of
                     variable
30B A9 00    LDA #0      ;pointer
                     table
                     and first
                     item must be
                     your
30D 91 69    STA  ($69),Y ;dummy string
                     variable
30F C8       INY
310 A9 02    LDA #2      ;NXTCHAR uses
                     ASCII with
                     bit 7 set
312 91 69    STA  ($69),Y ;but FP basic
                     with it clear
314 4C 39 D5$JMP  $D539$ ;so call
                     Applesoft's
                     own routine
                     to do it
                     ;(ends in RTS
                     for return
                     to Basic).
                  $$0D39 ;For FP basic
                     in Ram.
```

CLEARING HI-RES SCREENS

By Michael Mathison

*** NB THIS ONLY APPLIES TO APPLES (ITT version in the next issue)

As stated in the Applesoft reference manual (page 134), CALL 62450 clears the "current" high res. page to black and CALL 62454 clears the "current" high res page to the HCOLOR most recently plotted. The "current" high res page is, of course, what is in location (decimal) 230, so to clear any high res screen before displaying it the following could be used:

For HGR:

```
POKE 230,32:CALL 62450:HGR
```

and for HGR2:

```
POKE 230,64:CALL 62450:HGR2
```

which gives a true clear screen without the initial view of the previous drawingwhich then disappears.

To use CALL 62454 you should first set HCOLOR to the desired color. Then set the high res page as required. Then HPLOT 0,0 (or anywhere else) and then CALL 62454. However to display the results you will have to resort to POKES, so use pages 132 to 133 of the Applesoft reference manual.

e.g. HCOLOR=5:POKE 230,32:HPLOT 0,0:CALL 62454:POKE -16301,0:POKE -16300,0:POKE -16297,0:POKE -16304,0

which will clear the HGR page 1 to orange.

A cruder method would be :

HGR:HCOLOR=5:HPLOT 0,0:CALL 62454

CONTRIBUTORS GUIDELINES

These are what they say- guidelines, and not a firm set of rules. Let me stress first of all that the important thing is to get contributions in whatever form or style.

However, the more material that comes in already in the desired format, the easier the workload will be, and hence this article.

The preliminary layout of the journal will be two 40- character columns per page, each (usually) 64 lines long. Text will be fill- justified, and program listings presented in standard format- i.e 40 characters per line, and not de- spaced with the 'POKE33,33' command. When you enter and list a program, therefore, it should appear on the screen in an identical format to the printed listing- a useful aid, we hope, to de-bugging.

Except for program listings and material submitted in a 'print- ready' format, all contributions will be transferred to APPLEWRITER files for proofing and editing, and if it is possible to submit articles on disk in this format, this will obviously be ideal. (We hope to publish shortly an Applewriter 'Print to Screen' option that will aid contributors without printers in the editing of their contributions.)

Although the title of the article should be included, this will be replaced with a Letraset larger- character heading. All disks will be returned, and before publication a final draft will be sent to contributors for approval. Although the editorial staff can insert the necessary printer commands, you may if you have access to a printer or when our ScreenPrinter routine is published wish to produce draft copies, and the following example shows the required commands. This is shown first as it appears on the video screen during input/editing, and then the resulting print- out is presented.

We would welcome the comments of other user groups, with a view to agreeing standards, and in the meantime the Applewriter source- files for the articles in this journal are available on disk.

```
!rm39
!cj
A SHORT ARTICLE
---------------
BY A.N.AUTHOR

!fj
```

The Applewriter text- processing program works best where material is presented as a series of paragraphs of about 200 letters or five line- fulls each.

Each paragraph should be seperated by a blank line and this is achieved by pressing RETURN twice at the end of each paragraph.

As a general rule each punctuation mark should be followed by a space- as should dashes. Failure to do this can lead todisastrous results when the fill- justify mode is used.

For basic program lines a 40- character left- justified mode should be used:-

```
!lj

10PRINT "NOTE THAT PROGRAM LINES DON'T"
20PRINT " NEED A SPACE BETWEEN THEM"

!fj
```

Finally, to conclude this brief example of the Applewriter-text-processing-program note the effect of careless use of hyphens without spaces.

Continued on page 26

Continued from page 26

A SHORT ARTICLE

BY A.N.AUTHOR

The Applewriter text- processing program works best where material is presented as a series of paragraphs of about 200 letters or five line- fulls each.

Each paragraph should be seperated by a blank line and this is achieved by pressing RETURN twice at the end of each paragraph.

As a general rule each punctuation mark should be followed by a space- as should dashes. Failure to do this can lead to disastrous results when the fill- justify mode is used.

For basic program lines a 40-character left- justified mode should be used:-

```
10PRINT "NOTE THAT PROGRAM LINES
DON'T"
20PRINT " NEED A SPACE BETWEEN
THEM"
```

Finally, to conclude this brief example of the Applewriter-text-processing-program note the effect of careless use of hyphens without spaces.

BASUG & THE IAC

While advising of the updating of the BASUG address to the Post Box, we wrote to the International Apple Core asking how we could get hold of copies of 'Apple Orchard' and in particular what the cost was. The point was made that the IAC was not 'International', because it does not quote such prices to cover the non-American supply of the magazine. The group is also very much American biased, which is only natural since that is where its roots are. In reply the following letter was received from the secretary of the IAC, Joseph Budge.

"Your request that the IAC be truly international was timely. Now that most of the basic organizational work is completed we are studying what methods will best serve all our membership. The problems in making the organization international arise from both distance and cultural differences. Distance means that communications are ridiculously slow or hideously expensive. We quote a higher price for the Orchard, for example, to reflect the dramatically higher costs of overseas postage. Cultural differences mean, primarily, language difficulties. We recently received three diskettes of excellent software from a German club. How do we make the softwre usable to our English speaking clubs. Worse, how do we make it accessible to our Japanese clubs?

Continued on page 28

Pippin's Page ~~~~

Edited for younger readers by Vernon Quaintance

Welcome to the first Pippin's Page. This part of Hard Core is for younger readers. It will be what you make of it. What I hope to do is to publish here as many of your own programs, hints, tips and queries as possible. I cannot include anything which is not your own original work, since someone else would own the copyright and their permission would be needed to publish. This is not to say that we cannot use something with which you were helped by parents, teachers or friends.

If you would like to see your programs in print here, just follow these few simple instructions:

1. Make sure the program works.
2. Record it on disk or on both sides of a cassette.
3. Label the cassette or disk with your name and address.
4. Write out the following information on a sheet of paper:-
   (a) The program name, and a brief outline of what it does.
   (b) The language used ie. Applesoft or Integer Basic, or Machine code.
   (c) The system size (16K, 32K, 48K) on which the program was recorded and, for a machine code program, the memory location at which it loads and its length.
   (d) Any other information which is not included in the program but which is needed in order to run it properly.
   (e) Your name, address, telephone number and age.
   (f) An indication of your interests.
5. Send the letter and program to:-
   V. G. Quaintance,
   "Pippin's Page",
   50, Beatrice Avenue,
   Norbury,
   LONDON, SW16 4UN

Provided your program will run properly, I will add it to the **Pippin Library.** The best, and the unusual programs, together with your hints and tips will be used on Pippin's Page as soon as possible.

Anyone sending a program which is added to the Pippin Library will have their disk or tape returned with the whole month's Pippin offerings recorded on it. The really good programs will also be passed on automatically to the main BASUG library and you will receive credits in the usual way.

Hardware ideas, operating procedures, gadgets, etc can all be written about and, if published, will be rewarded.

In future issues, I want this page to be mainly yours. I will, however, try to offer advice on how to improve your programs, etc. Please write in with your problems as well; someone will almost certainly know the answer.

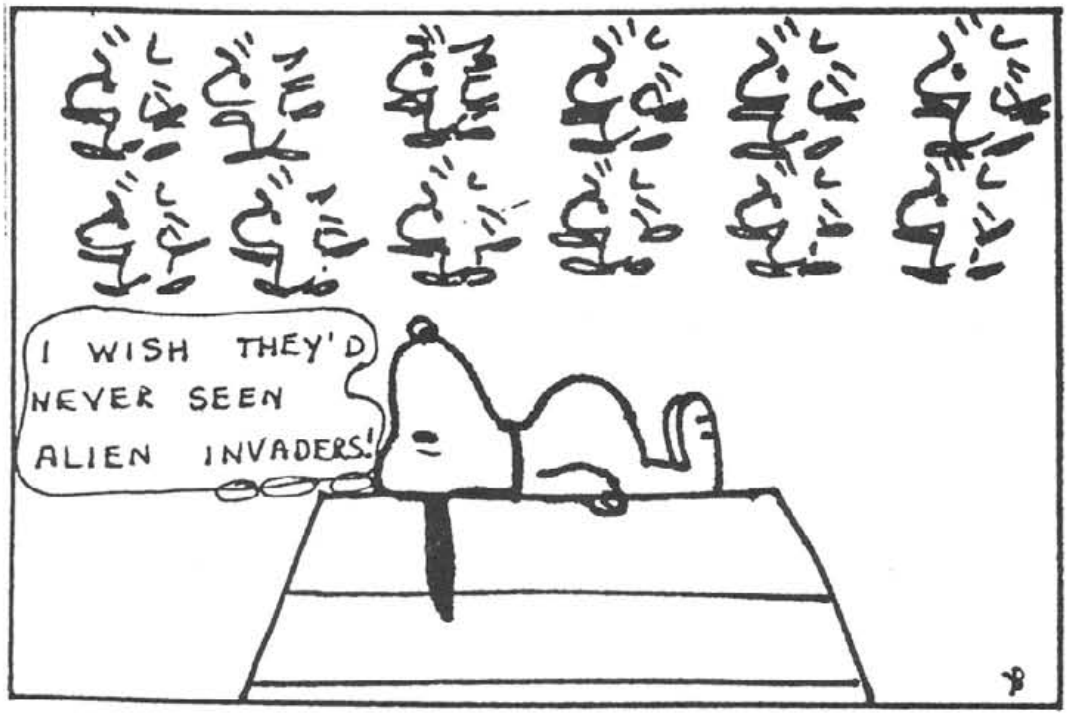
Pippin'sPage (continued)

I will leave you with a problem. Let's see the most elegant answer that you can devise. Assume that your latest games program in Applesoft starts at Line 100. It uses a lot of variables: numerics, strings and arrays, which must start off reset to zero or null strings. The game at present ends:-

```
3510      IF W = 1 THEN WON = WON + 1
3520      PLAYED = PLAYED + 1
3530      PRINT : PRINT "GAMES PLAYED = "PLAYED; "   GAMES WON = "WON
3540      PRINT : PRINT "DO YOU WANT ANOTHER GAME?  ";: GET K$ : PRINT K$
3550      IF K$ <> "Y" THEN PRINT : PRINT "GOODBYE" : END
```

Continue from 3560 in such a way as to reset all the variables, except the number of games won (WON) and the number of games played (PLAYED), and then re-run the game. You can also use lines 50 to 99 if you wish.

Happy Programming



Continued from page 26

I would greatly appreciate your thoughts on this matter. From your overseas vantage point you might have seen how other organisations dealt with these issues. Any suggestions you can make, either specific or general, would be gratefully recieved."

This problem is going to hit us soon. In one respect it already has. We have a disk and a magazine from a German club. Are you, for example, able to cope with a set of instructions in German. If not can you be sure you are not missing out on the program you want on that disk in the sotware library. What do we do? Let us know your ideas so that we can all get the best of all possible worlds.

John Sharp

EDITORIAL

COPYRIGHT & THE SOFTWARE LIBRARIES

One or two programs have come in that are marked as, or known to be, copyright. Obviously when we realise this they won't be accepted, but if any should slip through, and legal action should result; then the group would have to seek to show that it had acted in good faith, and thus pass the responsibility back to the member submitting the program. BASUG software, by the way, is not 'in the public domain' -copyright is held by the group and/or the various authors, and the software is supplied for member's personal use only. If you have friends who want our software, persuade them to join the group! A great deal of effort and expense goes into building and maintaining the libraries, and to make programs available to outsiders is to defraud your fellow members. (Sorry this has got a bit heavy, but these things ought to be put on record.)

CONTACT

We had written both to Microsense and to ITT. Microsense have now replied, and we are in the process of arranging a meeting with them. We are still waiting to hear from ITT, but expect to do so shortly. Hopefully by the next issue we'll have more to report on both fronts.

FEEL THE WIDTH

'Call-Apple' is probably acknowledged as being the premier Apple user-group magazine, having been established early in 1978. An analysis of a recent issue (October 1980) showed that, of the sixty pages, an equivalent of twenty-eight were advertising, giving 'real' contents of thirty-two pages. The first issue of Hardcore contained, on the same basis, thirty pages, and this issue about thirty eight pages. As to the quality ... well, you must be the judges of that.

CREDIT DUE

With the rush to get the last issue produced I didn't give any credit for the help received. So, in retrospect, thanks to Tony Williams for the production work, and Frank Kay of Lux Computers for the use of their Qume daisy-wheel printer. The majority of this issue (including this editorial) has been produced on my Centronics 737 printer using Apple-Writer files, my own lower-case adaptor, and Ian Trackman's 'Go-Between' - a printer module to enable all of the **Centronics 737** print options (changing between **character** sets, underlining, forward and reverse half line feeds, etc) to be called from within a text file, and to enable properly justified printing using the proportional-width correspondence quality character set. The quality of the resulting output can be seen, but in addition the ease of preparation was quite astonishing!

LOCAL CONTACTS

People are beginning to come forward as local contacts. At first these local groups will tend to be informal, with members meeting perhaps in each other's houses. As to the future, time will tell. We feel it's best to refer to local groups by their base town rather than any regional name, as with the rapid growth it's by no means clear yet just how local each group will be. Here's the list so far - lots of the country remaiuns to be covered:-

Frank Fletcher	Wilmslow 25204
David Row	Basingstoke 57541
Leo Crossfield	Maidenhead 35345
John Maltby	Eastbourne 2254
Len Gould	Sheffield *

* Len Gould is moving in mid-April,and his new phone number will be advised.

DISABILITY

We have had several letters mentioning this. 1981 is the International Year of the Disabled, and micro-electronics offer far-reaching possibilities of aid for the disabled. Several group members have already expressed their willingness to help in this field, but the first requirement is to gain an up-to-date picture of what's already happening. So please, if you're disabled yourself, or if you work with the disabled, or if you would like to help - write in with as much information as you can.

TIME WARP ?

A recent new member is Robin Hood (yes, that's really his name) who edits a game magazine called the "Herald" which caters for aficionados of Dungeons & Dragons, Wargames, Rail Baron, 1829, En Garde, etc. If any of these interest you, he can be contacted at 103 Oxford Gardens, London W10 6NF. Robin is in the process of reviewing the various adventure-type games available for the Apple, and we hope to publish his reports in due course. He is also going to look at the Eamon adventures, and give us an expert's opinion.

SOMETHING FOR NOTHING

Yes! - Members can advertise private sales and wants free in 'Hardcore'.

GO WEST

If you live near Bristol, you may be interested in 'BAUD' - the Bristol Apple Users and Dabblers. They meet once a month at Datalink's premises at 10, Waring House, Redcliffe Hill.

The next two meetings are on 9th April (Review of Business Packages) and 8th May (Interfacing) - both at 7.30pm.

SPECIAL OFFERS ON DISK DRIVES

There are several 5 1/4" drives available as alternatives to the official Apple drives, and very nice some of them look, with their own power supply, double density storage, etc. BUT - before you buy one, just make sure that you understand all the differences. The Apple Disk Operating System is copyright, and therefore isn't used by some of these alternative drives if they require their own interface cards. You may recall that there was a court action last year between Apple and the manufacturers of the ITT drives as a result of which, we gather, DDP (who now manufacture 'ITT' drives) switched to an I.B.M. format for their drives. Which is fine if you want to run a dedicated and self-contained business package, but not so good if you want access to off-the-shelf Apple software.

EVEN CHEAPER DISK DRIVES

Still on the subject of disk drives, Crofton Electronics of Twickenham are selling Shugart SA400 drives at about £140. These are the same drives as used by Apple, but of course they don't have the Apple electronics built into them. Does anyone know what's involved in converting these.

DOS UPGRADE PROBLEM

John Rogers, who looks after the Contributed Software Library, has a problem which is probably shared by many others - he has an early ITT Disk Controller Card which uses 24-pin ROMs (74LS474's) instead of the usual 20-pin ones. It seems the new 3.3 ROMs are not available in this format. Does anyone have a solution to this problem?

PROTOTYPE CARD

Vero are now producing a prototype card to fit into the Apple expansion slots. It retails at £9.60, but we can get them at £8.00 - cash with order please. An obvious use for these would be to install 2K of static RAM or a PROM into the unused address range C800 to CFFF. Anybody care to design a circuit and write it up for the next issue?

U.S.MAGAZINES

Several people have enquired about setting up magazine circulation groups. This would have to be done on a local basis because of the prohibitive cost of postage. The procedure would be that subscriptions are taken out for the desired magazines (about eight of them, I would guess) and the cost is shared by the participants, who each take out on loan one magazine at a time. If you are interested, either in organising this or in participating, please get in touch.

APPLE BOOKS

We gather that several of these are now being published. The Computer Graphics Primer is reviewed in this issue, and Adam Osborne is publishing an Apple II User's Book - 'Mine of Information' hope to have this in time for the North London Hobby Computer Fair at Easter, at which they are exhibiting. 'Pete & Pam' should also have two Apple books by the time you read this.

COMAL

This language is 'a structured and extended Basic, combining the ease-of-learning of Basic with the desirable elements of Pascal'. It's attracting a lot of attention, and Commodore are putting a PET version 'in the public domain'. Do any of you have any experience of Comal, or know of an implementation of it for the Apple?

HOW MANY APPLES?

I've seen an estimate recently that there are about 20/25000 Apple systems in the U.K. already, and the forecast turnover figures of Microsense would suggest that they will sell between 5000 and 10000 systems this year. Which gives about 30000 prospective BASUG members. In practice, I'd guess that between 5% and 10% of users will in fact join.

David Bolton

IAC NOTES

The following notes summarise some of the information filtering over from the INTERNATIONAL APPLE CORE.

The March Bulletin of the IAC mentions that Steve Wozniac - co-inventor of the Apple - had a plane crash recently. Both he and his fiancee are now OK after a spell in hospital.

DISK OF SOFTWARE

The first disk of software has arrived. It is mainly graphics, scientific and maths orientated. It is available by writing to the Software Library, or by asking at club meetings. It is disk No. 6. For catalog see UPDATE No 1.

APPLE ORCHARD

We have had no response to letters requesting how to get Apple Orchard via the IAC. All the information comes as a routine subscription order. As the price is not going to be much different, we suggest you subscribe direct, details as below. Not only will you get the magazine quickly but you can be sure of getting a copy, since we have missed the boat on issue

2; it is supposed to be sold out. We are doing our best to get hold of a number of copies of issues 1 and 2. The club will also take out a subscription to put a copy in the library.
To subscribe direct write to :-
APPLE ORCHARD SUBSCRIPTIONS
PO BOX 1493
BEAVERTON
OR 97075
USA
The subscription is $10 per year, for 4 issues, published quarterly, with overseas postage of $10 giving a total of $20. The Bankers Draft or Money order in $USA should be made ot to "THE INTERNATIONAL APPLE CORE ". You can get a Bankers Draft from your bank, but it will cost you about £3; you can get an International Money Order from The Post Office, again at a colossal fee. The cheapest way is to find a branch of BARCLAYS BANK INTERNATIONAL, which will make you out an International Money Order over the counter for about 60p.
IAC AGM
The AGM of the IAC is being held in Chicago on the weekend of the 2/3 May 1981, at the Marriott Hotel. If anyone is going to be in that area at that time would they let us know. It would be nice to have a representative there.
HAM RADIO SIG
The Ham Radio SIG meets every Sunday on 14.329 MHZ (8 PM East Coast Time), net control WB7TRQ located in Cheyenne Wyoming. They also run an exchange library.
PASCAL
In the States, Pascal 1.1 is reputed to be out. Registered users can get an updated version by sending $15 and their disks. The update plus manuals will be $65 from dealers.
The following details on the update have been supplied :-
Changes affecting overall operation

1.All files from 1.0 are included within 1.1 but are not necessarily interchangeable.
2.Text files with mope than 40 blocks must be divided using the old Editor prior to being used with version 1.1.
3.All read commands will work much more quickly with 1.1.
Version 1.1
1.Includes a system swapping option that allows you to maximize available memory space.
2.Allows you to shift the keyboard into lower case (like Apple writer).
3.Sets up consistent rules on uses of suffix.TEXT or.CODE.
4.Modifies the one drive startup.
5.Allows you to create Exec files.
6.Provides a SAVE command other than through the Filer.
7.Provides new editor prompts.
8.Includes expanded find and replace commands.
9.Adds optional automatic spacing and improves hyphenation.
10.Corrects and improves COPY FROM and WRITE commands and will no longer clobber your file when the buffer is nearly full.
Changes affecting user programs
1.Upper and lower case letters are interchangeable.
2.A new "V" option to check the VAR parameters of type STRING.
3.Program code file can contain up to 16 segments.
4.New "Next Segment " and " SWAPPING " options.
5.Automatic return to text mode on termination while in graphics mode.
6.Improved LIBRARY utility prompt lines.
7.A new UNIT called "CHAINSTUFF" to "chain" to another program.
8.LIBRARY.CODE used instead of FORTLIB.CODE.
PRODUCT NEWS
RAM CARD WARS -- With the number of RAM boards coming out, the LANGUAGE CARD is to be launched separately for about $250.
APPLE PILOT which uses Pascal 1.1 runs fast and has 4 editors - text, music, Graphics and character set generator. Worth a look, particularly by educators.
APPLE MODEM -- Apple is coming out with its own modem. Rumoured to have touchstone or pulse dialing capability and dial tone or busy signal detection. Numerous features above the D.C.Hayes modem.
NEW MOTHERBOARD -- The latest version of the Apple (revision 8?) allows for easy modification by you to use the shift key to get upper and lower case. (How you might do it isn't mentioned -ED). This could void warranties. There is also a socket for the 80 column Sup-R-terminal.
DOS BOARD -- Computer Data Services is marketing a DOS board to go in slot 0, thus giving you 10.5k more useable memory.
PROGRAM PROTECTION
There is a problem being created by the LOCKSMITH program. This package is basically an analog copy package, which allows copying of almost any disk. One exception is it will not copy itself. Software firms are threatening to withdraw ads from magazines which carry adverts for Locksmith. There are arguments both ways. The IAC is asking for opinions on this touchy piracy subject, and we invite yours

to pass on to them. The questions being asked are should the IAC:-
1. condone its sale and use?
2. advertise the Locksmith in The Orchard?
3. do nothing?
4. refuse ad space in the Orchard?
5. express an official position against its sale and use?

USING THE APPLE TO CONTROL OTHER EQUIPMENT

1 - Hooking up the APPLE to a CINE CAMERA

By John Sharp

There are many animation packages that allow very good movement at a very fast speed. For example Bill Budge's 3D animation package allows changes of up to 40 frames a second. This only works effectively if the object is very simple. As the complexity goes up then the calculation required means that the time between each frame lengthens.

There are times when it would be useful to study movement of very complex objects or indeed to use the computer to produce cartoon films. Although a machine code program can go fast, and can dramatically speed up BASIC programs, it would not help with a BASIC program that took a minute or more to generate a frame, since it is unlikely to speed it up sufficiently to give the standard 18 frames a second. The problem is not as crucial if only part of the frame is being altered, e.g. a man walking across the screen, since his shape is the only thing to be altered.

One approach would be to try to use VIDEO, but there are sync problems. Single frame causes all sorts of problems. There was an article in a recent " COMPUTER AGE " describing a video recorder with a camera for making animated films with the usual drawings on cells, but this cost £4000 for the camera and modified video recorder. If anyone has solved this problem cheaply there is money to be made not just with the APPLE but with other micros as well. It would seem that 80% of all animation is being done by computer, because of the high cost of getting the illustration done and also the time required. To bring the cost down to the level of £2000 or so would mean a tremendous advance and with software combined with The Graphics Tablet or The Versawriter, someone stands to make a mint. The advertising market in particular is moving to computer animation in a big way.

So what about the use of a conventional cine camera? From a cost point consideration from the amateur's point of view it would be 8MM - particularly at the present because the video boom has meant a drop in sales of cine cameras and the prices have dropped. There are snags however. The relative cost in terms of time, for example. About 8 mins film cost the same as a few hours of video tape. The tape is reuseable whereas the film is not. It is also a case of instant viewing with the video system, which can be important as regards checking exposures. There is also the difference in viewing conditions. The video system is not very portable, whereas the cine system is. But with no video single framing possible at present, this is all academic. So what does a cine system cost and what is involved?

THE CAMERA

A number of facts need to be considered in relation to the camera.

The most important one is the shutter speed for taking any photograph from a television screen. A TV picture is refreshed or changes every 1/25th of a second. If a photograph is taken with a speed greater than this then the familiar bar occurs across the picture because that part of the picture was not being generated during the time the shutter was open. The shutter speed of most SUPER 8 cameras is around 1/30th of a second. This would rule them out in normal circumstances; (I am using such a camera, as I shall describe, which gives me some extra flexibility, but means I have to resort to more complex control procedures). There are cameras with shutter speeds around 1/20th, normally in the ranges called XL, meaning low light cameras. They have large aperture lenses, normally F 1.2.

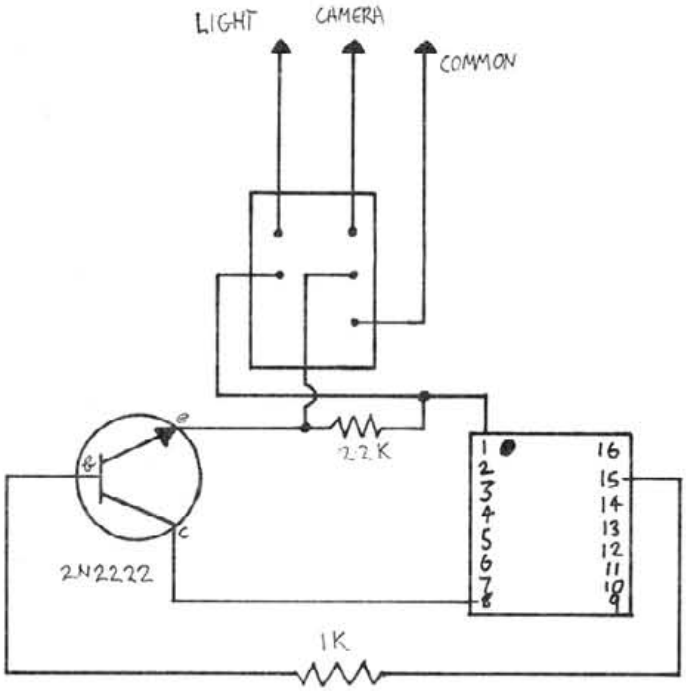
The next item for consideration is the viewing system. It is essential to have single lens reflex, since you will require to be working fairly close up and so need to ensure the screen is correctly seen in the viewfinder.

Then the camera must have some way of having the lens aperture fixed at a value you decide upon. It is not possible to rely on automatic exposure for the following reasons. If you have generated a small object and the rest of the screen is black, the camera (if automatically adjusting the exposure) would integrate the whole area and return a low level; it would then open the aperture and overexpose the object you are generating. As you bring more light onto the screen by generating more in the way of graphics then the integrated light level will have gone up and so the aperture will be closed down. This results effectively in a fade. How this is overcome will be described later.

The next consideration is, does the camera have a control which is mechanical or is it an electrical make and break. It is soon obvious that the latter is a must, for the camera needs to be switched on and off by the computer. When you consider that 18 frames are generated per second, 5 minutes of film requires 5400 separate pictures to be drawn. That is a lot of time to waste if you have to push the cable release by hand. It is possible to use a relay to push a mechanical release but it means a lot of construction.

You will also need to have a firm tripod as the system will need to be running for a long time in precisely the same relative positions.

So what camera am I using? I bought a EUMIG 881 PMA, on impulse rather than taking full note of the considerations I have laid down above. I did not check fully on the shutter speed, because the instruction book said it would do time exposures from 1/12th of a second to 1 minute ! What I did not do was to check if the single frame exposure shutter speed was less than 1/25th second. It is not - in fact it is 1/30th second. However, it is possible to have the exposure range described, but it requires a little bit of extra control from the computer. This exposure system works by integrating light levels over a period of time, and will allow exposures to be taken at night on single frame. This means I have to use a method of switching the shutter off by using a light rather than letting it go off after a fixed time as is the conventional way. The time can be made adjustable by a delay loop in the program on the computer, and can thus make fade in and fade out sequences easy and very controllable.



At this stage it would be best if I described how the Apple controls the camera and then go on to talk about the practicalities.

THE CONTROL CIRCUIT

I make no claims on the originality of the the circuit. It is taken from a circuit used to run a pair of cassette recorders in 'NIBBLE' No 4, p 35. The diagram of the circuit shows that it is very simple. The relay was bought from TANDY and appears to be one of the cheapest on the market at £1.25. There are two contacts, a make and break. The make is used to switch on the camera; at the same time the light, used as described below, is switched off. The switching is controlled by a POKE -16295,0 to operate the relay, and a POKE -16296,1 to reverse the original action, i.e. the camera is now no longer operating and the light is switched back on.

DETERMINATION OF EXPOSURE LENGTH

The normal camera exposure with the EUMIG is done automatically with a light meter through the lens, but this is not possible with the set-up I am using. I am using the additional facility for taking long time exposures whereby a light shines on another cell under the lens. This is meant to take very long exposures, i.e in a dark building. It works by integrating the light over a period of time up to a minute. If a bright light is shone on the cell at any time it will cause the shutter to be closed. This is how I am able to switch off the camera at any time I choose by having the APPLE cause the switching on of the bright light. Firstly I want the length to be at least 1/25th of a second for reasons outlined above. The sequence of exposure is as follows. The camera is initially set up with the auxiliary light on. When the computer comes to take a frame, a POKE -16295,0 causes the relay to be operated. This does two things; it makes contact so that the shutter of the camera is operated and it breaks the contact lighting the auxilliary light. The shutter is left open until a POKE -16296,1 causes the relay to cease to operate. When this happens the light comes on and the shutter is closed. It is common practice to take two shots of each frame. This cuts down the drawing, either by hand or computer and speeds up the actual filming; the reason for having so many frames is more to prevent flicker than to produce greater smoothness of motion. In order to make sure the light stays on for long enough to act (the computer can switch far faster than the bulb can light and dim), a small delay loop must be inserted.

I will assume that the required length has been determined, what happens is controlled by this program:-

```
10 REM THE PICTURE HAS BEEN GENERATED
20 TIME = 2500 :REM THIS HAS BEEN DECIDED UPON AS BEST
EXPOSURE
30 POKE -16295,0 : REM SWITCH ON THE CAMERA AND SWITCH
THE LIGHT OFF
40 FOR N = 1 TO TIME : NEXT
50 POKE -16296,1 : REM SWITCH OFF ,SWITCH LIGHT BACK ON
60 FOR N = 1 TO 1000 :NEXT : REM ALLOW THE LIGHT TIME TO
STAY ON LONG ENOUGH
70 REM EITHER TAKE ANOTHER FRAME OR GENERATE ANOTHER
FRAME
```

It now remains to know what is the best length to use in the timing loop. The only positive way to work is run a test film. A program was written to set the camera up and produce the test film. The sequence in running this is as follows:-

1. The camera is focussed and centered using the first options and then a test loop is run.
2. The shutter is set manually at F16 and locked.
3. The screen brightness and contrast on the TV or monitor screen are set to fixed values by the use of an normal exposure meter (e.g. Weston Master V) held up against a fully white screen. This allows repeatable results when you actually have to film.
4. Using knowledge of the film speed and the stop setting, it is possible to guess roughly the time needed.
5. The option exposure test is run and a number of frames run off at different times by inputing values of D when requested to do so by the program.

FILMING AND THE RESULTS

The same sequence of operations is run through with the setting up program before the film starts. The actual program to generate the animated film has the timing loop, similar to the short demonstration listing above, inserted into it.

I have made one brief sequence which drew a set of ellipses which changed from a vertical line, extended through a vertically squashed circle and then into a true circle, a horizontally squashed circle and finally into a horizontal line. This proved very successful, and the definition and smoothness of change was far more than I could get from any of the animation packages. It is going to be of far greater use in exploring other geometric work I am involved in, and it is also satisfying to know that I am using the APPLE as a tool and not a game playing toy.

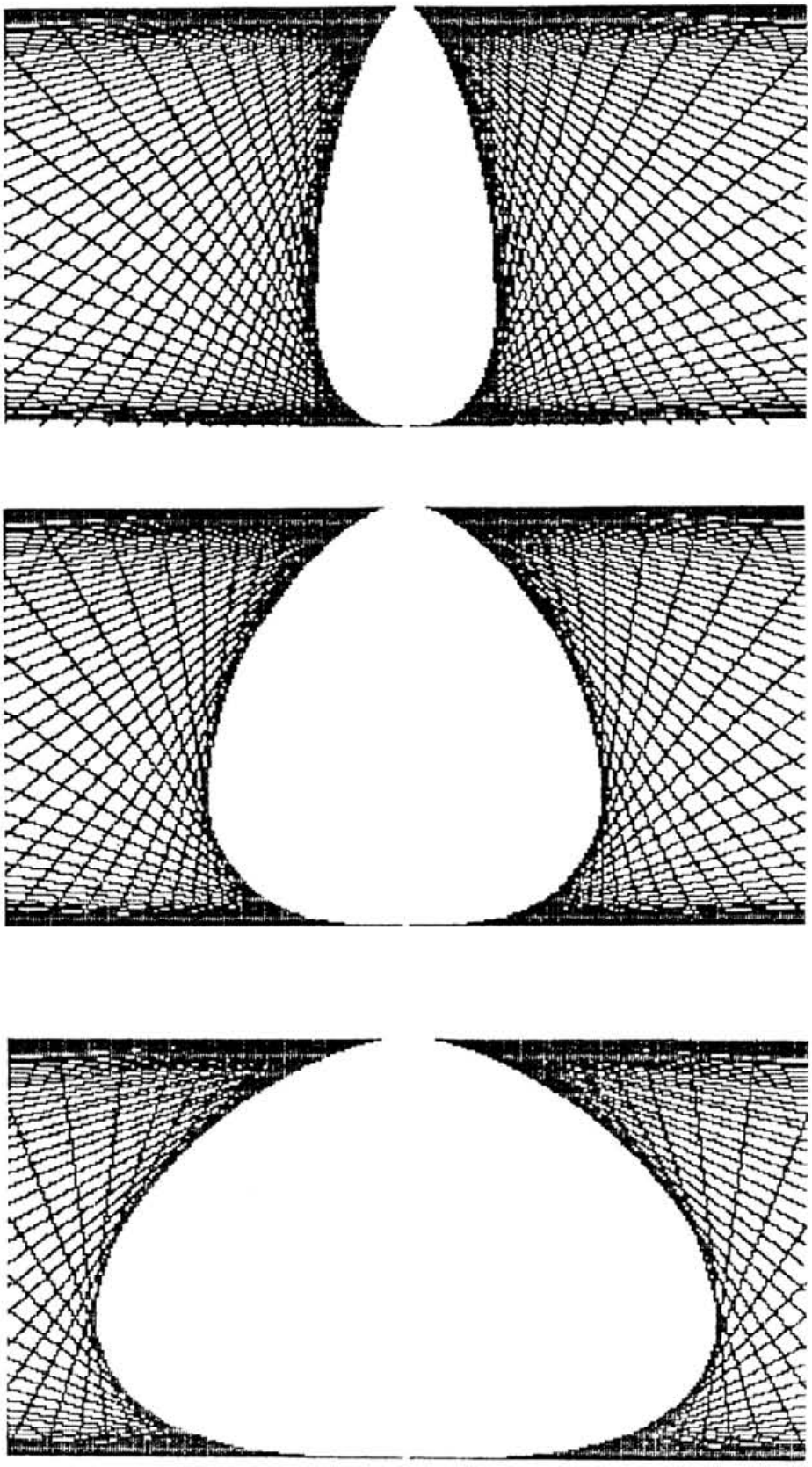
OTHER USES

There must be many other uses for switching that could use this circuit, or a group of them since I have only used one of the three possible.

It has also been suggested that the filming of text as screen-by-screen dumps could be carried out by this method. One would need very low grain film for this and my initial experience suggests AGFA is NOT the film to use.

```
50  POKE  - 16296,1: REM  xxx MAK
    E SURE SWITCH IS OFF
100  REM    $$$$$$$$$$$$$$$$
            $                $
            $  CAMERA SET    $
            $  PROGRAM       $
            $                $
            $  3/1/81        $
            $                $
            $$$$$$$$$$$$$$$$
110  TEXT : HOME
120  VTAB 3: PRINT "xxx CAMERA SE
    T UP TEST xxx
130  PRINT : PRINT "1.  CENTERING

150  PRINT "2.  FOCUS
155  PRINT "3.  EXPOSURE METER CH
    ECK "
158  PRINT "4.  EXPOSURE TEST LOO
    P "
159  PRINT "5.  CAMERA
160  PRINT "6.  EXIT
165  PRINT : PRINT "AFTER A TEST
    HIT ANY KEY TO RETURN  TO  M
    ENU "
175  PRINT : PRINT : PRINT "HIT N
    UMBER TO GET TEST
176  GET A
180  ON A GOSUB 500,700,1000,1100
    ,1500,1600
200  GOTO 110
250  GET A$
490  REM  xxx CENTERING
500  HGR2 : HCOLOR= 3
510  HPLOT 0,0 TO 279,0: HPLOT  TO
    279,191: HPLOT  TO 0,191: HPLOT
     TO 0,0
520  HPLOT 0,0 TO 279,191
530  HPLOT 0,191 TO 279,0
540  HPLOT 139,0 TO 139,191
550  HPLOT 0,95 TO 279,95
```

```
580  FOR N = 1 TO 3: PRINT  CHR$
     (7): NEXT : REM xxx BELL
585  GET A$: RETURN
690  REM  xxx FOCUSSING
700  HGR2 : HCOLOR= 3
780  FOR N = 0 TO 139 STEP 4
790  HPLOT 139,0 TO 139 - N,95
795  HPLOT 139,0 TO 139 + N,95
800  HPLOT 139,191 TO N + 139,95
810  HPLOT 139,191 TO 139 - N,95
820  HPLOT 0,95 TO N,0
830  HPLOT 279,95 TO 279 - N,191
840  HPLOT 0,95 TO N,191
850  HPLOT 279,95 TO 279 - N,0
860  NEXT
870  IF ZQ > 10 THEN  RETURN
880  FOR N = 1 TO 3: PRINT  CHR$
     (7): NEXT : GET A$: RETURN
900  NEXT
990  REM   xxx CLEAR SCREEN TO
     WHITE FOR EXPOSURE METER
      CHECK
1000  HGR2 : HCOLOR= 3: HPLOT 1,1
     : CALL 62454
1010  FOR N = 1 TO 3: PRINT  CHR$
     (7): NEXT : REM xxx BELL
1020  GET A$: RETURN
1090  REM  xxx EXPOSURE LOOP TEST

1100  TEXT : HOME
1110  PRINT "EXPOSURE LOOP TEST "

1120  PRINT : PRINT "THE LOOP IS
     AS THE FOLLOWING LINES"
1140  PRINT "IT ALLOWS STOPWATCH
     TIMING OF THE       CAMERA"
1150  LIST 1170 - 1192
1160  VTAB 18: PRINT : VTAB 18
1170  INPUT "VALUE OF D  ";D
1175  IF D < 1 THEN  RETURN
1180  POKE  - 16295,0: FOR N = 1 TO
     D: NEXT N: POKE  - 16296,1
1190  REM   xx -16295,0 IS SWITCH
      ON,-16296,1 IS SWITCH OFF
1192  REM  xxx USE D<1 TO RETURN
       TO MENU
1195  IF ZQ > 10 THEN  RETURN
1200  GOTO 1160
1490  REM  xxCAMERA TEST EXPOSURE

1500  HGR
1505 ZQ = 15
1510  GOSUB 780
1550  VTAB 22: GOSUB 1170
1557  IF D < 1 THEN  RETURN
1560  FOR BM = 1 TO 36
1565  VTAB 22: HTAB 30: PRINT BM
1570  GOSUB 1180
1572  FOR WS = 1 TO 1000: NEXT
1575  NEXT
1580  GOTO 1550
1600  END
```



STILLS FROM THE FILM 'METAMORPHOSIS OF EGG CURVES'

DIFFERENCES BETWEEN PALSOFT AND APPLESOFT HI-RES GRAPHICS

By Richard Teed

There have been a number of articles on this topic but none have been complete; this one is hopefully more complete.
As many, if not all of you know the sizes of the APPLE and ITT screens are different: they have the same number of horizontal lines (192) and the same number of words per horizontal line (40) but the size of the words is what causes the change in proportions. The APPLE has an eight bit word consisting of seven low bits plotting plus a non plotting high bit which is used for colour. The ITT uses nine bit words with the first eight bits corresponding to those on the APPLE except that bit eight plots and is not used for colour: the ninth bit comes from a latch on what was the button on games paddle three. Location C05E (decimal 49246) acts as a switch to enable bit nine, this will cause bit eight to be copied into it. Location C05F acts as a switch to disable bit nine and when toggled will prevent bit eight being copied into bit nine. If you have an ITT try this:

```
100 HGR:POKE-16302,0:FOR N=8192 TO
16383:POKE N,255:NEXT
```

You will see that all the screen is set to white except for the ninth bit of each word. Now try this:

```
100 HGR:POKE-16302,0:POKE 49246,0:FOR
N=8192 TO 16383:POKE N,255:NEX
```

As you can see all the screen is white because bit nine has been enabled. As you can imagine because of the complicated way in which the ninth bit is set and cleared all high res routines will be slower than those of the APPLE, 1.5 to 2 times slower infact.

The odd way in which bit nine is set also means that you cannot BSAVE the screen because bit nine will not tag along, so you will need a special packing and unpacking program for the bit nines when saving high res screens to disk.

As the dots on the APPLE screen are square the two extra plotting dots per word of the ITT cause displays to be compressed to get them back to normal, all x co-ordinates must be multiplied by 9/7. When you try to plot on the APPLE, multiple plot statements of the type:

```
100 HPLOT X1,Y1 TO X2,Y2 TO X3,Y3 TO
X1,Y1
```

Are legal, but will give a SYNTAX ERROR on the ITT, which must have a separate plot statement for each line.

There are also differences with shapes on the two machines. If you have a shape table on the APPLE the following program will put shape one on the screen.

```
100 HGR:XDRAW 1 AT 100,100
```

Because the shape is placed on the screen by doing an EXCLUSIVE OR between the screen and the points making the shape up, the shape is made visible irrespective of the setting of HCOLOR. On an ITT however if HCOLOR is not set to a non black value nothing will appear with an XDRAW.

All the above problems are shown very well by the program COMPUTER BISMARCK (which however can be converted to the ITT with very little trouble).

We now come to the final difference between the machines: that of the collision counter which is totally different for each machine and can lead to fatal errors.

On the APPLE the collosion counter is incremented by one for each point of a shape that is placed on a pixel of the screen that is set to one, but only when using the draw command. For the XDRAW command the reverse is true: the collision counter is set to the number of pixels of the shape set to the on position and deccremented by one for each one that corresponds to a screen pixel set to on.
The ITT is of course much more convoluted. The DRAW and XDRAW commands both behave as the DRAW command does in the APPLE but the ninth bit makes it almost unusable.

For bits one to eight of the screen the following applies: For each pixel of the shape set to one that corresponds with a screen pixel set to one, the collision counter will be incremented by one. For the ninth bit the following applies: If any pixel of the shape set to one is on a ninth bit of the screen and that bit of the screen is off then the collision counter is incremented by one. If the bit on the screen is also set to a one then the collision counter is incremented by two.

Using the above rules on an ITT to detect collisions between shapes is something you will find very difficult to program for.

These as far as i know are all the differences between the Apple II and the ITT 2020 as far as high resolution graphics are concerned. If anyone knows of any others, then please let us know.

SPEEDING UP YOUR BASIC

By Graham Rubens

Most of us are keen to write faster programs at one time or another. I recently found a very simple way of finding out what is the fastest way to tackle a problem with several possible solutions.
The Apple has a simple speaker which is toggled by accessing memory location -16336 decimal to produce a single click. The faster you toggle the speaker, the higher the pitch of the sound.
If you want to test several methods of acheiving a similar result you simply write a little program to try each in turn about 30 times. Toggle the speaker during each pass and listen to the resulting noise.
The higher the tone, the faster the method. Here is a sample program to give you the idea. :-

```
 10 S=-16336 : REM SPEAKER LOCATION
 20 A = 1 : A% = 1 : M = 30 : REM TEST ARGUMENTS
 30 FOR X = 1 TO 30
 35 IF A = 1 THEN Y = PEEK(S)
 40 NEXT X
 45 REM FIRST TRY
 50 FOR X = 1 TO M
 55 IF A = 1 THEN Y = PEEK(S)
 60 NEXT X
 65 REM SECOND TRY
 70 FOR X = 1 TO M
 75 IF A% = 1 THEN Y = PEEK(S)
 80 NEXT X
 85 REM THIRD TRY
 90 FOR X = 1 TO M
 95 IF A% = 1 THEN Y = PEEK(S)
100 NEXT
110 REM FOURTH TRY
120 FOR X = 1 TO M : IF A = 1 THEN Y = PEEK (S)
130 NEXT
140 REM ANY FASTER ?
```

By using this simple technique I was able to find the fastest alternative from my different methods. Incidentally for the advanced programmer, when all your Loops are tested and you have a listing of the whole program ready to strip out all rems, try deleting all the variables from your NEXT statements. (i.e. FOR X = 1 to 30 : FOR Y = 1 to 50 : A$(X,Y)="": NEXT : NEXT). This makes programs very hard to read but a little faster. DON'T FORGET TO KEEP A FULL LISTING !

READER'S LETTERS

[redacted],
14th March.

Dear David,
I return the User Group disc, which must be corrupt as it is not possible to run your adventure program as it just crashes.
I would be quite happy to act as representative for Yorkshire or whatever and believe I could get quite a few interested people together. Let me know what is involved.

Len Gould

‡‡‡‡‡‡‡

[redacted],
19th March.

Dear John,
Many thanks for the Introductory Tapes.
I have just finished adapting a program to run in Palsoft called MICRODOC. It was originally published in PCW in Feb,'79; but was 'bugged' and had unsatisfactory screen formatting.
It is used to establish voltages, currents, and wattage ratings applying in complex D.C. circuits, and is really a design tool for Electronic / Electrical Engineers. If a tape, listing, and full documentation is likely to be of use to the Software Library I would be pleased to arrange this.
I have urgent need of manuals specifically for the ITT 2020 - the hardware is different from the Apple II in that there is an extra 4116 chip for the mysterious 9th Bit, and the PCB layout is totally different. This makes fault diagnosis and location very difficult when one has Apple manuals only.
Whilst on the subject of ITT Hi-Res graphics I take issue with you re your recent article in Hard Core. ITT screen width is 0-359 (i.e.360) and Apple II 0-279 (i.e.280). This gives a ratio of 9/7, not 9/8, and using 9/7 on my Hitachi monitor gives the expected true circle.
I have not yet figured out how to access the extra bits on the ITT in machine code. Any clues?

Frank McLaren

‡‡‡‡‡‡‡

[redacted],
Tyne & Wear.

Dear Keith,
Please help a grey-haired, bleary-eyed gibbering idiot who has spent so much time underground, in MICROSOFT ADVENTURE, that the moles have sent a petition to the Local Cave Authority to have me put.
I have reached the status of Adventurer Class C with 262 points from 260 moves and feel loth to ask for assistance but find I have run out of ideas.
Problem- How to get the chest from the Pirate assuming it's the fifteenth treasure.
The least bit of assistance as possible please.

Yours adventurously,
T.W.Paddon.

‡‡‡ We asked Andrew Margolis of Lion Micro-computers for help, as he is a Grand Master Adventurer, having scored the maximum 350 points. He was surprised, not to say amazed, that you were stuck at this particular point, as the Pirate would have told you where he was to be found. Review your progress so far, considering every word, and you should find the clue you need.

‡‡‡‡‡‡‡

[redacted],
9th March.

Dear Sir,
I have some software which may be of interest to you and I will be pleased to know how you wish to receive this, i.e. do you supply the necessary discs?
At present I also have two expensive programs: 1,Computer Ambush, 2.Napoleonic Wars, which I personally consider not to be very good value. However, if any other group member wishes to borrow these programmes in order to make their own assessment before purchase, I am quite willing to loan these.

Yours faithfully,
R.J.Davies.

‡‡‡ To keep administration down, we ask you to send in programs on your own disks. If you're ordering software from the library, you'll automatically get your disc back. If not, a note saying 'Please Return Disk' in big letters will do the trick.

‡‡‡‡‡‡‡

[redacted],
8th March.

Dear David,
Thanks for your excellent magazine and disk.
I have been a member of Call-Apple for two

years and have long wished that we had an Apple club here. Well the best things are worth waiting for. I wish you and BASUG all the best.
I must sing the praises of Computer Micro Works in Ohio. They manufacture a DOS Switch which plugs into the disk controller piggy-back fashion. It uses one of the old 3.2 PROMs and both the new ones. There is also a little switch which sticks out of the back of the Apple. All I now have to do is flip the switch and re-boot in 3.2 or 3.3, no need for a BASICS disk. The only problem is that it is slightly fiddly to fit, because of the PAL card in slot 7, but it is definitely worth every penny of the thirteen pounds that it costs. I ordered mine by 'phone with my Access card and received it in eight days.
Finally I'm having trouble with DOS 3.3 and the Mountain Hardware Copyrom in the Romplus board. After using the Catalog function I cannot save any programmes to disk. It gets as far as entering the name of the programme and then crashes into monitor. The only way to recover is to switch off and on again. This doesn't happen when DOS 3.2.1 is up. I have read that there were a couple of bugs in the DOS but my local supplier 'dos' not know of any. Is there a bug? And are there any fixes?

Yours sincerely,
Robert Crossley.

†††††††

[redacted]

5th February.

Dear Mr Bolton,
Thank you for sending me membership information. If sounds as though the group is in its embryonic stage. What's the chance of setting uo more local meetings, say like London, or South-East London.
Our machine is partly for businees and partly for pleasure. This means that one interest we have is being able to provide an emergency back-up system on a reciprocal basis. Have you found others with a similar requirement?
We have experience in using four brands of accounting package - Computech, Mitron, Systrek, and Vlasak. Their quality and price vary considerably, and yet no one seems to have done a Which? type report on such things.
Has your group come to an opinion as to which of the popular magazines feature Applesystems more than the others? Or indeed would the group consider encouraging any magazine to give more prominence to Applesystem information.
One problem I'd like solved is a mailing list with membership list capability, i.e:-

-sort in order of surname, although address labels must begin with Mr, Mrs, Rev., etc.
-include details such as membership number, class of membership, etc.
-print out membership lists with all information on one line.
-print out a list of only those members in a given classification.

Unfortunately Apple Post falls appreciably short of this. Any ideas?
Finally, the group may be interested in knowing I'm helping my employer, Balfour Beatty, set up a modem link between an Apple in Saudi Arabia and a DEC PDP11in our London office. Have others experience in this, especially spurious echoes when routing via satellite telecomms?

Yours sincerely,
Paul Vernon.

††† Re the mailing list problem, I think that AIM from 'Nibble', which is in the Contributed Software Library, does most of what you want

†††††††

[redacted]

8th February.

Dear 'BASUG'
Many thanks for my introductory disk and copy of 'Hard Core'. I thought that most of the programs on the disk were marvellous and included some I had been struggling with. Equally I found 'Hard Core' most useful, particularly the section on machine code programs. I couldn't get 'Life' to do anything other than plot lo-res points - I obviously need some instructions. Equally I don't understand the object of 'Haunted Cave', or what I'm supposed to do.
As my small contribution to the library I enclose my version of 'Hangman' which differs from the version on the intro. disk in that two players take turns playing rather than one playing the computer; and 'Space Landing' - a common program concerned with piloting a lander onto the surface of the moon, etc.

Re Apnotes, I see that one item in the contents is 'Applesoft Out of Memory'. I sometimes get this error in the middle of programs - I got it with 'Zombie Island' when I first tried it but now it seems to work O.K. Is it something to do with HIMEM and LOMEM commands - I confess I don't really understand these.
Next can anyone tell me what the Integer commands AUTO, DSP, and MAN mean (I understand MOD).
Finally I think that BASUG is a fantastic help to beginners like me. Please keep up the good work.

Yours sincerely,
Anthony Freedman.

The There are article on both 'Life' and 'Haunted Cave' in this issue, and also an explanation of Integer Basic commands. Can someone explain the 'Out of Memory' problem - this would make a useful article for the next issue.

#######

[illegible],
16th March.

#######

[illegible],
[illegible],
11th March.

Dear Sirs,
Please enrol me for the Pascal course - I can bring my own hardware including monitor and language system. I would appreciate a recommendation as to where I could book for Bed & Breakfast on Saturdays 16th & 30th May.
I had offered the loan of my C.J.M. Microbox, joysticks, and graphics software for the group's stand at the North London Computer Fair; but find that I will not be in London until after the show. However if any member is likely to be in the Manchester / Stockport area before the show, I will be only too pleased to loan this equipment and collect it when I come down for the Pascal course.
I offer a short review of HI - RES MYSTERY HOUSE by ONLINE SYSTEMS INC. This is an extremely well-written piece of 'games' software which uses the HI - RES capabilities of the Apple to full advantage and provides quite a challenge to the adventurer.
The facility to 'save' the state of the game has been provided for and is very useful as it took several colleagues and myself many hours to finally 'crack it'. The game entails entering the usual commands (e.g. Look, Take, Get, Move, Go, etc.), the object being to move throughout the different rooms and secret passages in the house hoping to kind and kill the person who is murdering the other occupants, and ultimately leaving the building laden with jewels. There are 'notes' in certain rooms which can be read giving clues and several dead bodies are to be found. The graphics are extremely good and certainly add to the entertainment value of the program. A clue! The towel is useful in locating the jewels.
Incidentally, a friend of mine has written a routine, which he has no objection to being published, to convert the 'Slideshow' to run in Pascal. The access is incredibly quick, and one feels that with Winchester drives (to store many frames) it may allow interesting experiments in animation.
Keep up the good work! A user group of this quality has been long overdue.

Yours Sincerely,
Barrie Holmes.

#######

[illegible],
24th February.

Dear Sir,

Please find enclosed disk holding my contributions, which I hope may prove useful to other members.
May I congratulate you on the first issue and the disk? These alone are worth the subscription fee!
An Adventure question - my son has been playing Mystery House & has found the treasure. He also knows that ######## is the killer. He insists there is more to it than this, is there?

Yours faithfully,
Eric Sausse.

Not that we know of!

#######

Glascow,
13th March.

Dear Martin,
Congratulations on the successful growth of your Group down there.
A few wee thingwhich the resources of your club might be able to help with:-
1.Let's get a register of those of us who have modems.
2.What about Radio Hams who can transmit data on 80 metres?
3.I have Mbasic running on the Apple and also on a 64K Altos. The Apple documentation for the CP/M card details "up load" and "down load" to take CP/M programs down from }un}another}uf} machine to the Apple. What I want to do is to get the Apple to talk directly to the Altos but however hard I have tried nothing happens - help!
4.I have the IBM / CP/M compatibility disk from Tele Systems. What I want to do is to take 160 byte records from IBM System 32 down to the Altos - and possibly also to the Apple. For some reason the transfer program does not work if the IBM data record exceeds 128 bytes. This is of the greatest concern to me since, in one of our office applications we have 50000 160-byte records and I am desperate to get these on to an Altos hard disk by, at the latest, mid June - Help!

Yours sincerely,
Colin O'Hara

††† Colin runs 'Apple-Eden', a member group of the International Apple Core.

†††††††††††††††††††††††

CHAIRMAN'S CORNER

The group is now beginning to take off, but it is still very much in a state of flux. There has not been a great deal of feedback as to what everyone wants so do let us know; maybe you are too busy programming.

I have learnt a great deal about not only the APPLE but also about the wide range of expertise and interests the members have. There have not been too many problems with the machine, just problems with members realising the potential and not having the time or expertise to put them all into practice.

The past few months have taught me quite a lot about the Apple. When you have to act as a source of information you have to look at things closer than you would have otherwise. There were many aspects I hadn't covered, which necessity made me look at and find they were not as difficult as they appeared. It is just a matter of finding the time and putting the effort in. It is suprising how simple it is to use the machine when you do put the effort in to study it properly.

I have also learnt that the ITT 2020 is not the machine to buy if you want to use graphics, although I will qualify this in a moment. Consequently I have changed from being an ITT owner to being an

By permission of Johnny Hart and Field Enterprises, Inc.

Apple owner. I would like to take this opportunity to explain why, and would welcome any reply from Apple (UK or USA) and ITT.

First let me say that they are both good machines. The Apple has two obvious faults in case design. The switch at the back is very poor quality; I am continually hearing of them breaking. So now I always switch off at the mains. The lid is perfectly designed to act as a lever to allow you to ram it down onto the mother board as you take it off. Why couldn't they have done as ITT did and give put some practical thought into it. The flatter ITT surface is better as well for standing disk drives and monitors on.

There is one other thing that I do not like about the Apple, namely The Autostart Rom. It may give you faster cursor movement and the better I,J,K,M movement on pressing the ESC key only once, but these are standard in any Editor package. The autostart / turnkey system is good for the non programmer, but if you hand over control to the machine you lose control. There are all too many times when the system is completely lost and the only way to recover is to switch off. All your program is lost; and if you haven't saved it, it can be very frustrating. Combined with the faulty switch, a very bad fault.

However, although ITT have given better resolution graphics, it is not significant enough when you actually look at the display. The ninth bit problem doesn't cause any problems if you are writing all your own programs in BASIC, but if you want to use Apple programs, you could be in trouble. This is especially obvious with software like Bill Budge's Space Album and Apple World, and the Versawriter and Graphics Tablet.

There were problems with the disk drives over copyright, which was settled out of court. This may have made ITT hesitant over committing themselves too much. I spoke to Ken Mace, in ITT marketing, last February who promised a new reference manual would be ready in March. This is a year ago I and I still haven't seen one. They may not be aiming at the personal market, but they do not appear to be aiming at anyone else either; they seem to be doing little else but manufacturing the machines. Perhaps they would care to put their point of view in the next issue of 'Hardcore'.

But having thrown brickbats at both ITT and APPLE, I must say I have only too much admiration for the machine as a whole. The quality is best summed up in an episode I witnessed at the Personal Computer World Show in September last. While I was talking to someone on one of the stands, a girl brought some coffee along. There was an Apple which was lying with its' lid off and as luck would have it, one cup went all over the mother board. The machine was switched off and mopped up. A considerable amount of work went into cleaning it up. But after all that it still worked, and to the best of my knowledge is still. Whether the owner knows its history or not I don't know.

I may have the reservations I outlined above, but I have many more about other machines. However, if you get even a tenth of the enjoyment out of your Apple or ITT 2020 that I have then you have little to complain about either.

PRINTER INTERFACING

By Chris Murphy

The intention of this series of articles is to cover the common types of printer interfaces available for the Apple, to explain in what I hope are simple terms the basis on which each operates, and therefore give some guidelines as to which type of interface to use having decided upon a particular type of printer. The interface cards which I will examine are the Apple Parallel Printer Card, the Apple Centronics Card, the Apple High Speed Serial Interface Card, the Apple Communications Card, the C.C.S. Serial Card and one or two others in passing.

Firstly, let us quickly examine the types of printer available. These fall into three main categories, electrostatic printers, thermal printers and impact printers. The latter may be further sub-divided into dot matrix and daisy wheel type printers.

Electrostatic printers require special aluminium coated paper, and tend to be somewhat disparagingly referred to as "cooking foil" printers. Most of those available for the Apple (e.g. the Axiom series of printers and plotters) come with a special interface, and therefore there is no choice as regards interface.

Thermal printers, such as the Silentype, again require special stationery, and are usually supplied with their own special interfaces.

Most impact printers, be they low cost 80-column dot matrix or high quality high price daisy wheel, have either serial or parallel interfaces (or in some cases both!). Serial interfaces may also be refered to as RS232C or CCITT V24, whilst parallel interfaces may also be referred to as Centronics-type interfaces.

The interface is there to provide a means of passing information from a computer to a peripheral device and to deal with translating the information from a form which the computer recognises to a form which the peripheral understands. The two types mentioned above are the major types of data transfer. The distinction between them is fairly simple. Parallel interfaces transmit information simultaneously down parallel data lines, whilst a serial interface transmits information bit by bit down a single wire. That is the overall view - in practice, it's a bit more complicated than that, and I shall be going into more detail on both, starting with the parallel interface. The next article will deal with serial interfaces.

PARALLEL INTERFACES

We mentioned above that a parallel interface transmits data down a number of wires at the same time. To take a very simple illustration, let us suppose that we wished to print the letter "A" on the printer. As far as the Apple is concerned, it recognises A as ASCII character 65(decimal) or 41(hex). This is converted into its binary equivalent(01000001) and each bit is transmitted down a separate wire. A requirement of parallel data transfer is that all data bits must be both transmitted AND received simultaneously. Therefore, the interface has to handle two tasks : firstly, it must make sure that the computer and the printer are working in harmony, so that when the computer is ready to send information, the printer is ready to receive it ; and secondly, it has to handle the conversion of data from character format to bit format for transmission. The major limitation of the parallel interface is that

the length of the cable connecting the computer and the printer is limited; the longer the cable, the more signal distortion occurs.

The Apple Parallel Printer Card consists of a plug-in board with a ribbon cable and a jumper block. Also included is a very useful manual. As the parallel card is designed to interface with a variety of printers, the user has to customise it by supplying and wiring up a connector for the printer and by wiring up the jumper block to suit the particular printer chosen. The ribbon cable has twenty wires, but in most cases only ten to twelve of these need to be used. Wires 1 and 20 are "circuit" or "logic" grounds whilst wires 10 to 17 should be connected to the appropriate data input pins in the printer. The commonest types of plug used are 25-pin Molex connectors or 36-pin Amphenol connectors (if you are not sure what these are, I would suggest that you obtain a copy of RS Components catalogue). Wire 2 is the ACK signal, whilst wire 8 is the STROBE. The jumper block must be configured to suit the printer's "Handshaking". This is the general term used to describe synchronising the computer and the peripheral device. The way this works with a parallel interface is that when the computer is ready to send information, a character is transmitted to the interface. The interface then sends a STROBE signal to the printer and waits until it receives ACK back from the printer before attempting to transmit another character. Most parallel printers require either positive or negative STROBE and either positive or negative ACK. The Apple Parallel card manual gives wiring diagrams for the four possible combinations. However, you should ensure that the printer you are using has a manual describing its own interface.

That is a brief summary from the hardware viewpoint; now let us consider the parallel interface from a programming viewpoint (N.B. - in all these articles, I will only be looking at Basic). In order to access the printer, you use the command PR£n , where n is the slot number in which the parallel card is located. The parallel card has a number of default options, of which the most important are that it assumes 40 columns output width and will automatically add a line feed after a carriage return. Thus, each line sent to the printer, say on a program listing, is terminated by a carriage return and a line feed.

The Apple parallel card uses Control-I (ASCII 09) as a control character, and in order to accomodate different types of printers, the card has a number of inbuilt commands which it recognises whenever it receives Control-I. The most important are summarised below :-

CTRL I nN - Turns off monitor screen and prints n columns per line on the printer. The number of columns may be between 40 and 255

CTRL I I - Outputs to monitor and the printer.

CTRL I K - Turns off automatic line feed after carriage return.

CTRL I CTRL <letter> - Changes printer control coomand character to specified letter.

CTRL <letter> CTRL I - Puts things back to normal.

An example of using some of these commands in a program would be :-

```
10 PR#1 : REM TURN ON PRINTER
20  I$=CHR$(9)  : REM SET PRINTER CONTROL
CHARACTER
30 PRINT I$;"80N" : REM SET PRINTER
WIDTH TO 80 COLUMNS
35   PRINT "THIS WILL APPEAR ON THE
PRINTER BUT NOT ON THE SCREEN"
```

```
40 PRINT I$;"40N" : REM SET LINE LENGTH
BACK TO 40 COLUMNS
45 PR#0 : REM DISABLE PRINTER
50 PRINT "HERE WE ARE BACK ON SCREEN"
```

Apple recommend three tips to guard against printing problems. These are :-

1. Before issuing a PR#n command, be sure to home the cursor and clear the screen (CALL-936 or HOME in Applesoft will do this).

2. When printing more than 40 characters per line, always reset the line length to 40 before using PR#0.

3. Before listing a program on the printer that has printer control commands embedded in it, change the control character from I to something else ; and of course, don't forget to change it back to I before running the program!

The Apple parallel card can also be used as a general purpose 8-bit parallel output card to drive such devices as music synthesizers and digital to analogue converters. This is detailed in the manual.

The other type of parallel card produced by Apple is the Centronics Card. This is the parallel card supplied with a jumper block ready wired for Centronics printers and an Amphenol connector on the ribbon cable. Unlike the normal parallel card, it does not generate a line feed after a carriage return, as most Centronics printers already do this. If you are buying a printer which has a "Centronics type interface", then you may be able to save yourself some soldering and avoid buying a connector by getting the Centronics card, but do make absolutely sure beforehand that the printer has the same interface as a Centronics printer with regard to handshaking and the wiring of the connector.

Finally, there is one other parallel interface which I have used ; the Computech Diplomat Parallel Interface, which is supplied by Computech together with the Microline 80 Printer. This gets round some of the programming problems when using the Apple interface by incorporating most of the options on switches on the inerface card. Thus , for example, setting the line length is done by switches, and all the program needs to do is issue PR# commands to select and de-select the printer.

- - - - -

So much for a brief introduction to Parallel printers. In future articles, I will be covering Serial interfaces.

BEGINNERS' PAGES

By John Sharp

GETTING IT RIGHT ON THE SCREEN

There are many little things that add polish to a program. There is nothing so disconcerting to me than to have a program run and the first line come up half way down a page. The listing or whatever has gone before has not been removed and it is difficult to know what you should be reading. It is simple to have the first line clear the screen and set the cursor up to the top left. This is easily accomplished by having the first line of your program:-

```
10 TEXT:HOME
```

(In Integer Basic, of course, you would have to use 'CALL-936' in place of 'HOME')

This also helps if you have been editing using 'POKE 33,33', because it sets the text window how it should be.

Whereas it is easy to put text into programs, it is not always easy to have it laid out the way you want it to read. One of the most common errors I have seen is the overrunning of a word from one line to another, so that it is split. The effect is something like:-

```
THIS IS AN EXAMPLE OF HOW A LINE THAT SO
MEONE HAS TRIED TO FIT ONTO THE SCREEN L
OOKS
```

Not only is it difficult to read but it takes some of the polish from your program. How can it be overcome? Well the easiest way is to look on the screen. If you are typing a BASIC line in for a PRINT statement then you open quotation marks to begin a string. When you come to the next line as soon as the string comes in line with the quotation marks you would be printing on the next line when you actually ran the print statement. So leave a few spaces if the word is going to go past the place where the quotation marks sit on the line above. This all sounds difficult when you are reading it so by way of an example :-

```
10 PRINT "THIS IS HOW TO GET A VERY AWKW
ARD LOOKING STATEMENT PRINTED ON THE SCR
EEN "

20 PRINT "THIS IS HOW TO GET A VERY AWKW
ARD      LOOKING STATEMENT TO PRINT COR
RECTLY BY LEAVING A CORRECT NUMBER OF SP
ACES."
```

If you ran these two lines then the first would have part of the looking on one of the screen lines and part on the next one. Line 20 would be all right. There is a snag if you try to edit, using 'POKE 33,33' as in the last issue. The best way then is to run the line when you have altered it and go back and insert a few spaces to make it come right.

FINDING OUT WHERE A MACHINE CODE PROGRAM ON A DISK HAS BEEN PUT INTO MEMORY.

The APPLE has to know what it is doing with all the information it is processing and so it has to keep some sort of list of where things are, such as where a program is in memory, or where it has stored the variables in a program. Such items in a list are called pointers. When you have loaded a machine code program for example there are some locations in memory that know exactly where it went. We can use these and some others to find out the length, in order to know for example how to save the machine code to tape.

The two locations that contain the information where the program starts are (in hex) AA72 and AA73 and the locations where the length of the programs are stored are AA60 and AA61.

Now when the first microprocessors were designed there was a specific request from the customer that if a two byte number was stored in two successive locations, it was done the wrong way round. That is that the low byte (often called the least significant byte or LSB) was stored first and the high byte (the MSB or most significant byte) was stored second. This continued to the later processors for no logical reason. This means we have to be careful when we interpret the values of the numbers in the two locations. As ever, an example is far easier to understand so let us take one. Suppose you want to look at the starting address of a binary file. From BASIC, BLOAD the program and then go into monitor with a CALL -151. Now list the two locations AA72 and AA73 by typing :-

```
AA72.AA73      and press RETURN
```

This will result in the contents of these two locations being displayed like this :-

```
AA72- 03 08
```

This tells you that the program starts at the hex location 0803 (remember! - reverse order).

Similarly if you list the contents of locations AA60 and AA61 by typing :-

AA60.AA61 and pressing return

and you get the result

AA60- 00 08

this tells you that the program is 0800 bytes long.

Now if you want to load the program onto tape then you must add these two together. It is easy to make A MISTAKE as I have done on the Introductory Disk in the Binary Copy Program. If you add the two numbers together you must take one from the result. Why ? Well suppose the length of the program is 5 Bytes and starts at 800 hex ; the program lies in the locations 800, 801, 802, 803 and 804, i.e. in 5 locations, the last one being 804 = (800 + 5) - 1. It is not too serious a problem, since it is better to have too much of the program rather than too little.

AFTER POKE 33,33 IT'S POKE 33,28

If you list a number of commercial software programs you often see a REM statement in a nice little box like this :

```
10 REM  ************************
        *                      *
        *  THIS HELPS TO       *
        *    DRAW  YOUR        *
        *    ATTENTION         *
        *  MORE THAN THE       *
        *   USUAL  REM         *
        *    STATEMENT         *
        *                      *
        ************************
```

It takes a lot of work to get it right because having typed the line in, when you list it (remember, it's the listing rather than the printing that you want to look right in this case) the monitor alters the spacing, which is why editing requires POKE 33,33 as described in the last issue. However, Michael Mathison discovered by accident that if you POKE 33,28 then type in the box as you want it, when you list normally (i.e. having set the text window to normal either with a POKE 33,40 or by typing TEXT) it will come out just as you want it to.

HELP !!!!

If you have any problems or any solutions which would be suitable for this column, please send them to the editor. After all we were all beginners once, and still are in some respects because no one can know everything about all aspects of programming.

BASIC DIFFERENCES

By John Rogers

The Apple computer supports two types of BASIC interpreter, one is Floating Point BASIC, usually called Applesoft or Palsoft (depending upon machine) and the other is called Integer BASIC. The main difference between the two is that Floating point BASIC can accommodate numbers with a floating decimal point, whereas Integer Basic, as the name implies, can only deal with integer numbers. This can be of use in programs that can produce non integer numbers but where only integers are required.

In addition there are a number of commands that are only available when using Integer BASIC,and these are, AUTO, MAN, DSP, and MOD. I shall only deal with these four commands as a specific question has been asked on their use. There are many commands that are available to Floating Point BASIC that are not available to Integer BASIC.

The command AUTO enables you to enter BASIC statements without line numbers, the machine will do this automatically. To put the machine into auto-line number mode, type for example

AUTO 100,10. The first number will be the first line number in the program and the second number is the line number increment. Before you run or list a program using the auto-line number, type CTRL-X, if you do not you will get a syntax error. To get out of auto-line number, type CTRL-X MAN.

To help debug programs, Integer BASIC has a useful command called DSP. This enables you to examine the contents of a particular variable as the program is run. For example if you wish to check on the contents of two variables called MIN and MAX, type DSP MIN and on another line DSP MAX then run the program. A display (hence DSP) of each of the variables will appear on the screen every time they are encountered during the run of the program.

The MOD command goes some way to help resolve the lack of floating decimal point in Integer BASIC. The easyest way to explain what it does is to use an example, such as the one below:-

```
10 LET X = 125 MOD 12
20 PRINT X
```

The variable X will return with the value of 5, ie. divide 125 by 12, which gives 10 5/12ths, the MOD command will give the fractional part or REMAINDER.

THE ADVENTURER'S FRIEND

By Keith Jones

Several people have asked just what "ADVENTURES" are. Basically an Adventure is a computer simulation of a real-life or fictional situation, where the computer, based on decision tables and probabilities determined by the author of the Adventure, works out and reports on the consequences of your instructions.

Most of the basic ideas in all adventures are developments from the various board game 'fantasy and fiction' series, often "Dungeons and Dragons" (T.M.), which have a very large following. The general idea is to have a fantasy character who travels into fictional worlds fighting monsters, rescueing princesses, pillaging towns, plus the rest of the everyday chores - O.K.,it's escapism - and very addictive too!

There are several broad varieties of Adventure:-

1. The original (Crowther & Woods) Adventure. This has 150-odd rooms, most of which you can wander around, collecting treasures and artifacts. Many of the rooms have "puzzle" entrances - this is a feature of many adventures and one which many computer owners find fascinating. The appeal lies in solving puzzles, problems, riddles etc. You also encounter monsters, creatures, etc. I hesitate to give further details for fear of revealing too much to potential players! Many players find this version of Adventure addictive - you can save the game to disk, and start again from the same point in this and several other adventures.

2. Puzzle Adventures. Scott Adams has written a classical series of these - each adventure has a theme - sci-fi, Dracula, Pirates, etc.

These are very much problem solving adventures since you have to complete a given task in these - often within a time limit. Like most adventures, these use a two word input - verb and noun - so you may come across a locked door and you could "use key" or "pick lock" to enter the next room / cave. The only disadvantage of these games is that once the problems have been solved you have exhausted the game (since the format remains constant). Scott's programs are very popular - they are written in assembly language and are well structured (from the games players point of view). Other similar adventures have been written by Programmers Guild and Greg Hassett are are of similar quality but have not been so well known. (Incidentally, Scott Adams' Adventure Hints book is now available, and so are his 'hints sheets' - the latter seem better value!)

3. Graphic Adventures. A new(ish) development - these are

high-resolution graphics adventures where the player gets either:-

a) In the case of Odyssey, Temple of Apshai Hellfire Warrior, etc., a map or series of maps showing location or terrain, monsters, treasures, etc.

or b) In the case of the Online Systems programs, high-resolution pictures of each room (a room being the adventure term for each different place or location - could be a spaceship, cave, castle, room, up a tree, etc.), sometimes with moving graphics. Again you have problems to solve or princesses to rescue or zombies to avoid, but the addition of graphics adds very greatly to the enjoyment of adventures. Titles available in this series are:-

0 Space Adventure
1 Mystery House
2 The Wizard and the Princess

The full-colour graphics of 'The Wizard and the Princess' are amazing! (and it's also a very difficult adventure). I think the next step in adventures will be a combination of the type of graphics seen in Sirius Software's space games but applied to the fantasy world of adventure.

4. English-Speaking Adventures - ZORK. This latest text adventure has many features that will endear it to seasoned adventurers - I'm already haggard playing it! In Zork (sub-titled 'The Great Underground Empire) you find a world to explore and conquer, and although much of Zork seems based on older adventures, the terrain is much, much larger and instructions are not limited to two words - that's right! - Zork will understand a command such as "Take the sword and put it in the cupboard". I'm just waiting for someone to bring out a high-resolution version next!

5. The Wonderful World of Eamon. As part of the group's software library, these are very much recommended as an introduction to the world of adventure. You will find all the elements of mystery, excitement, and magic in this series of adventures. An interesting feature is the ability within the series to build your own adventures by using the "Dungeon Master" module and answering a series of questions. You can change locations, treasures, monsters, etc., and build a world of your own with problems for other people to solve!

pascal pages

```
PROGRAM PASCALPAGE;     (* AUTHOR: FRANK KAY *)

(*$I-,S+ *)

USES APPLESTUFF,TURTLEGRAPHICS;

CONST   GREETING='WELCOME TO PASCAL PAGE!';
        SALUTATION='GOODBYE FOR NOW!';
        NQ='NOTE';      PQ='PROGRAM';
        EQ='END';       ZQ='PUZZLED';
        SEP=', ';
        NTQ='WHICH NOTE?';
        BADANS='ERROR!';
        NPROGS=1;

TYPE    PPNOTE=   TEXT;
        OPTION= (NOTES,PROGRAMS,DONEPP,PUZZLED);
        SEGNAME=PACKED ARRAY [0..19] OF CHAR;
        SEGS=    ARRAY[0..NPROGS] OF SEGNAME;

VAR     WORKING:        PPNOTE;
        WANTED,DONE:    BOOLEAN;
        HEADQ,A:        STRING;
        SCOUNT,COUNT:   INTEGER;
        PROG:           SEGS;
        CH:     CHAR;

SEGMENT PROCEDURE ETCHASKETCH;
```

```
CONST   BL='BLACK ';     WH='WHITE ';
        GR='GREEN ';     VI='VIOLET';
        OG='ORANGE';     BU='BLUE  ';

TYPE    SIMPLECOLOUR=SET OF SCREENCOLOR;

VAR     L,I,P0,P1   :INTEGER;
        CH      :CHAR;
        BACK:   SET OF CHAR;
        GND:        SIMPLECOLOUR;
        BACKGND,FOREGND,DRAWGND:  SCREENCOLOR;
        DRAW:   BOOLEAN;
        S:      STRING;

PROCEDURE SHOWDRAW;
BEGIN
PENCOLOR(NONE);
VIEWPORT(194,279,0,191);
MOVETO(194,50);
IF DRAW
THEN WSTRING('ON ')
ELSE WSTRING('OFF');
VIEWPORT(0,191,0,191);
MOVETO(P0,P1);
END;

PROCEDURE SHOWCOL(C:    SCREENCOLOR);
BEGIN
CASE C OF
BLACK:  WSTRING(BL);
WHITE:  WSTRING(WH);
GREEN:  WSTRING(GR);
BLUE:   WSTRING(BU);
ORANGE: WSTRING(OG);
VIOLET: WSTRING(VI);
END;
END;

BEGIN

BACK:=['X','W','G','B','O','V'];
GND:=[WHITE,BLACK,GREEN,VIOLET,ORANGE,BLUE];
REPEAT
REPEAT
PAGE(OUTPUT);
WRITELN('WHAT COLOUR BACKGROUND?');
WRITELN;
WRITELN('YOU CAN HAVE: (X) ',BL);
WRITELN('              (W) ',WH);
WRITELN('              (G) ',GR);
WRITELN('              (B) ',BU);
WRITELN('              (O) ',OG);
WRITELN('              (V) ',VI);
WRITELN;
  WRITE('PLEASE CHOOSE (');
READ(CH);
UNTIL CH IN BACK;
CASE CH OF
'X':    BEGIN
        S:=BL;  BACKGND:=BLACK;
        END;
'W':    BEGIN
        S:=WH;  BACKGND:=WHITE;
        END;
'G':    BEGIN
        S:=GR;  BACKGND:=GREEN;
        END;
'B':    BEGIN
        S:=BU;  BACKGND:=BLUE;
        END;
'O':    BEGIN
        S:=OG;  BACKGND:=ORANGE;
        END;
'V':    BEGIN
        S:=VI;  BACKGND:=VIOLET;
        END;
END;
WRITELN(') ',S);
FILLSCREEN(BLACK);
PENCOLOR(NONE);
DRAW:=FALSE;
DRAWGND:=BACKGND;
FOREGND:=BACKGND;
VIEWPORT(194,279,0,191);
MOVETO(194,30);
WSTRING('PEN');
MOVETO(194,60);
WSTRING('DRAWING');
MOVETO(194,90);
WSTRING('BACKGROUND');
MOVETO(194,20);
SHOWCOL(FOREGND);
MOVETO(194,80);
SHOWCOL(BACKGND);
VIEWPORT(0,191,0,191);
FILLSCREEN(BACKGND);
SHOWDRAW;
GRAFMODE;
REPEAT
P0:=PADDLE(0);
FOR P1:=1 TO 3 DO;     P1:=PADDLE(1);
IF BUTTON(0)
THEN    BEGIN
        REPEAT
        PENCOLOR(NONE);
        VIEWPORT(194,279,0,191);
        IF FOREGND=BLUE
        THEN FOREGND:=WHITE
        ELSE FOREGND:=SUCC(FOREGND);
        UNTIL FOREGND IN GND;
        MOVETO(194,20);
        SHOWCOL(FOREGND);
        VIEWPORT(0,191,0,191);
        MOVETO(P0,P1);
        FOR I:=1 TO 100 DO;
        END;
IF BUTTON(1)
THEN    BEGIN
        IF DRAW
        THEN DRAWGND:=NONE
        ELSE DRAWGND:=FOREGND;
        DRAW:=NOT DRAW;
        SHOWDRAW;
        FOR I:=1 TO 100 DO;
END;
        PENCOLOR(DRAWGND);
        MOVETO(P0,P1);
        PENCOLOR(SUCC(BACKGND));
        MOVETO(P0,P1);
        PENCOLOR(BACKGND)
        UNTIL KEYPRESS;
        READ(KEYBOARD,CH);
        TEXTMODE;
        WRITE('ANOTHER? ');
        READ(CH);
        UNTIL CH<>'Y';

        END;

        PROCEDURE CHECK;
        BEGIN
        WRITELN('TYPE THE LETTER CORRESPONDING');
        WRITE('TO THE ONE YOU WANT, OR <RETURN> ');
        READ(CH);
        IF CH IN ['A']
        THEN WANTED:=TRUE;
        END;

        FUNCTION ASK(X: STRING):        OPTION;
        VAR     T:      STRING;
                DONE:   BOOLEAN;
                ANS:    OPTION;
        BEGIN
        REPEAT
        DONE:=TRUE;
        PAGE(OUTPUT);
        WRITE(X,' ');   READLN(T);
        IF T=NQ
        THEN ANS:=NOTES
        ELSE    IF T=PQ
                THEN ANS:=PROGRAMS
                ELSE    IF T=EQ
                        THEN ANS:=DONEPP
                        ELSE    IF T=ZQ
                                THEN ANS:=PUZZLED
                                ELSE DONE:=FALSE;
          UNTIL DONE;
        ASK:=ANS;
        END;

        PROCEDURE ASKS(VAR X: STRING);
        BEGIN
        WRITE(X,' ');   READLN(X);
        END;

        BEGIN

        PROG[0]:='ETCHASKETCH          ';
        HEADQ:=CONCAT(NQ,SEP,PQ,SEP,EQ,SEP,'OR ',ZQ,'?');
        DONE:=FALSE;
        WRITELN(GREETING);

        REPEAT

        CASE ASK(HEADQ) OF
```

```
NOTES:  BEGIN
        A:=NTQ;       ASKS(A);
        A:=CONCAT(A,'.NOTE');
        RESET(WORKING,A);
        IF IORESULT <> 0
        THEN WRITELN(BADANS)
        ELSE
        BEGIN
        COUNT:=0;
        WHILE NOT EOF(WORKING) DO
        BEGIN   READLN(WORKING,A);
                WRITELN(A);
                COUNT:=COUNT+1;
                IF COUNT=20
                THEN    BEGIN
                        WRITE('PRESS <RETURN> TO CONTINUE ..');
                        READLN(A);
                        COUNT:=0;
                        END;
        END;
        WRITELN('THE END.');
        END;
        END;

PROGRAMS:       BEGIN
                WANTED:=FALSE;  COUNT:=0;       SCOUNT:=0;
                WHILE (NOT WANTED) AND (COUNT<NPROGS) DO
                BEGIN
                WRITE(CHR(ORD('A')+COUNT),': ',PROG[COUNT]);
                COUNT:=COUNT+1;
                SCOUNT:=SCOUNT+1;
                IF SCOUNT=20
                THEN CHECK;
                END;
                IF (NOT WANTED) AND (SCOUNT<>20)
                THEN CHECK;
                IF WANTED
                THEN    CASE CH OF
                        'A':    ETCHASKETCH;
                        END
                ELSE WRITELN('THAT''S ALL!');
                END;

PUZZLED:BEGIN
        (*      If all this seems unintelligible,
        then we suggest that you come and
        discuss this page with the Pascal Group
        at a BASUG meeting!  And you might consider the
        Pascal Courses we are starting!  The idea
        of this page is to publish Pascal Programs,
        swap Ideas, and provide a means of documenting
        Bugs and 'Tricks of the Trade'.

                There are two types of information
        which may be submitted for inclusion here.
        The first is a Note, which is a Pascal Text
        file with the name <anything>.NOTE.  This may
        contain any information at all, and this program
        will display any Note available for your
        perusal.  (for those of you not familiar
        with Pascal, you might like to read this
        program and see how it works!)  The second
        sort of submission is a Program.  These should
        be documented by means of a comment at the
        head of the Program, stating what System
        Resources the program requires.  As far as
        is possible, we will add these programs to
        this one, as Segment Procedures, so that they
        may be run from this program.  It is hoped
        that we can build up a good base of information
        to assist both the Student and the Experienced
        user of Pascal, and thereby encourage the
        wider use of the Language.  No contribution
        is too small!   *);
        WRITELN(chr(18),'Not for too long, we hope!');
        WRITELN('To see what Pascal can do,');
        WRITELN('Try a Note or a Program.',chr(20));
        SCOUNT:=60;
        FOR COUNT:=12 TO 24 DO NOTE(COUNT,SCOUNT);
        END;

DONEPP: BEGIN
        WRITELN(SALUTATION);
        DONE:=TRUE;
        END;
END

UNTIL DONE;

END.
```

APPLE 6502 ASSEMBLER/EDITOR

A REVIEW

by Ian Trackman

Having toiled for many long hours with a motley assortment of 6502 Assembler / Editors, none of which really provided all of the facilities needed for writing advanced machine-code programs, I was delighted to hear that Apple themselves were publishing the "official" Apple Assembler / Editor. I knew that a prototype had been in existence for some time and I was eager to try out the fully-debugged "all singing, all dancing" production version. Sadly, I was disappointed. The Apple Assembler falls below the high standards of programming and documentation that we have come to expect as a matter of course from Cupertino.

The system consists of two main modules, the Editor and the Assembler, linked together by a Command Interpreter. Source files are created using the Editor and then assembled by the Assembler directly to disk. The benefit of this method, compared with co-resident Assembler / Editors which assemble their source-code and store it in RAM, is that the maximum amount of memory is kept available for the source code. There is about 30K of free RAM, which should be enough for about 1500 lines of source-code - quite a respectable chunk of program - and supplementary source-code files can be "chained" during assembly. Coupled with this, the Assembler will handle labels of up to 250 characters in length, making it potentially a very powerful program.

Outside of testing and debugging a program, most of my development time is spent in writing and improving the source-code in the Editor and it is therefore essential to have a comprehensive range of editing facilities and pseudo-opcodes at my disposal. It is here that the Apple package shows up it weaknesses. To its credit, the Editor has most of the commands that I would have expected - ADD, COPY, DELETE, FIND, INSERT, LIST and REPLACE - together with a full single-line editor, which has the useful ability to present for editing all lines containing a designated string. The first letter of a command is sufficient to invoke it and most commands will work on a selected range of program lines. In addition there is REPLACE, which is a combined DELETE and ADD but there is no MOVE command. MOVE is simply COPY and DELETE and I shouldn't be expected to do this two-step operation myself when the Editor could help.

The SEARCH (and REPLACE, which is linked to it) is painfully slow. For instance, it takes about one minute (!) to search through a large source-file for a three-character target word. That's abysmal, particularly when compared with say, the "Search and Replace" in Apple-Writer or, putting modesty aside, my own Super Editor, which takes around two seconds to do the same thing. A minor criticism - the "clear to end of page" should happen immediately after the SEARCH command has been given and not wait until the first target line has been found.

The LIST command displays program lines together with their line-numbers, which is useful for editing, and there is also a PRINT command, which omits the line-numbers and formats the source-code into columns on the screen, which makes it easier to read. However, if you spot something during a PRINT which you want to edit, you then have either to do a FIND to locate its line-number or LIST the code again. I would like to be given the current line-number when I interrupt a PRINT in progress. Another complaint about LIST and PRINT is that Apple say in the Manual that the tabbing, which you can change during editing, has been set to the "standard column positions for 6502 assembly language". Unfortunately, the screen, being only 40 columns wide, can't cope with such a display, so that every time I use the system, I have to type "TABS 8,11,18" to prevent wrap-around. I would like to be told in the Manual where to find the tab settings so that I can change them on a permanent basis. Also, when listing, I would like to be able to scroll upwards as well as downwards through the source-file.

The Editor uses the blank space as its tab field delimiter but it can't differentiate between that type of space and a space in a string constant, ASCII data or comments. As a result, although the code assembles properly, the Editor output can be confusingly displayed. It also means that you mustn't put spaces in the names of chained files. The Editor and the Assembler also treat the tabbing of comment fields after single-byte opcodes (CLC, PHA etc.) differently - the Assembler adds an extra field.

Whilst on the subject of information in the manual, there is a memory map but the pointer addresses are not given. These are needed, for instance, when transferring source-code created on another assembler and stored by it as a binary-file. (I worked out that the HIMEM pointers are in $E and $F, which are the important ones if you want to SAVE a BLOADed

file). Incidentally, Apple decided to save the source-file to disk as a text-file, rather than as a binary-file memory dump, which means that loading and saving is slow - typically, over one and a half minutes to load a big source-file from disk.

There is a bug in the EDIT routine. If you delete or truncate the tail-end of a line, you can temporarily lose the next line. It is still in memory and will show up when listed but it can disappear when editing a range of lines.

The Assembler will not handle lines which consist only of a carriage return, although the Editor allows you to enter such lines, which can easily happen if you enter a new line before the Editor has finished re-locating the source-code after the previous line. Again, Apple-Writer could give this Editor a lesson in speed. You can also accidentally enter a control character into a line (since Control D and Control Q are used to exit from the ADD and INSERT commands) and I would like to see them as flashing characters when listing the code, rather than waiting to find them as errors during assembly. Inconsistently, the exit from the EDIT command is Control X.

So much for the editing facilities - now for the pseudo-opcodes. A good selection is supported, including BGE, BLT, DCI and the conditional assembly directives DO, ELSE and FIN. There is no macro-assembly capability. The Manual does not explain the symbol table codes which the assembler gives during conditional assembly and it would have been helpful to have fuller explanation, with examples, of the DSECT, DEND, EXTRN and ENTRY directives. Quite correctly, the Manual makes no attempt to explain machine-code programming itself but internal system commands are a different matter and need to be thoroughly documented.

Two bad omissions are the ability to use binary constants - very useful in masking opcodes - and mathematical manipulation of labels. I think that the latter may be a bug, since the Manual gives the impression that it can be done. For example, a line such as :-

```
LENGTH EQU #<FINISH-#<START
```

causes an assembly-time error.

Contrary to the convention used in most assemblers, Apple use < to denote the high-byte and > to denote the low-byte of a 16-bit address. I prefer them the other way around and I have altered the Assembler to suit my choice (the relevant addresses are $1C8E and $1C97).

The Editor uses MSB ON and MSB OFF instead of single and double quotes to indicate the parity setting of ASCII data.

The DW and DDB psuedo-opcodes will ausiness package, but not so good if you want access to off-the-shelf Apple software.

USING 2716 EPROMS ON THE MOTHERBOARD

By David Bolton

One of the early Apple Technical Notes described a procedure to accomplish this, and for reference that note is reproduced here.

The procedure as described suffered from several disadvantages:-

a) It required a high degree of expertise with a soldering iron, as well as a considerable amount of courage!

b) It voided your warranty.

c) As described, it was only applicable to ROMs D0 and D8 (the two not used on an Integer machine).

A better solution, on all three counts, is described below; but first let me describe the differences between the 2716, which is an Erasable Programmable Read Only Memory, and the 9316's used by Apple which are Read Only Memories, mask-programmed - which is to say that the code is programmed in during manufacture and cannot be subsequently changed.

Both chips use the same pins for the Address Lines (Pins 8 thru 1, 23, 22, 19); the Data Lines (Pins 9 thru 11, 13 thru 17); and Ground (Pin 12). The remaining pins are used differently, however. On the 9316 Pin 24 is +5volts; Pins 20 and 21 are Device Selects, Active Low; and Pin 18 is Device Select, Active High. On the 2716 Pin 21 is +5volts; Pins 20 and 18 are Device Selects, Active Low; and Pin 24 is Device Select, Active High. Thus, the assignment of Pins 18, 21, and 24 differs.

(A brief diversion here on the Apple's use of the Device Selects. Pins 20 and 21 of each ROM are commoned and each one is connected to the Chip at location F12 on the board, which selects the correct ROM depending on the Address Line. Pins 18 of all on-board ROMs are commoned, and held high by a 3.3k resistor to the +5v line. If this pin is pulled low by the INH signal (from a peripheral card) then all on-board ROM should be disabled.)

Now for the procedure itself, which involves no alteration to the motherboard or indeed to the EPROM. You will need a suitably programmed EPROM, a 24-pin socket - one with short tags about 4mm long, not a wire-wrap one (This will cost about 40 pence); and a couple of inches of thin flexible wire - I use 5-amp fuse wire.

The first stage is to remove the ROM you wish to replace, and put it aside for safekeepng (you'll want to put this back before having any warranty repairs done - remember, any unauthorised alteration voids the warranty, and this at least in theory includes both these EPROMS and the cheap expansion RAM that everybody uses). Note the orientation of this ROM as you remove it - a small semi-circle signifies the Pin-1 end, which on the Motherboard will be at the bottom (i.e. towards the keyboard).

Next, take the socket and bend over or break off pins (or tags, if you like) 18 and 21 - refer to the diagram, but note that this shows the view from above, and not below.

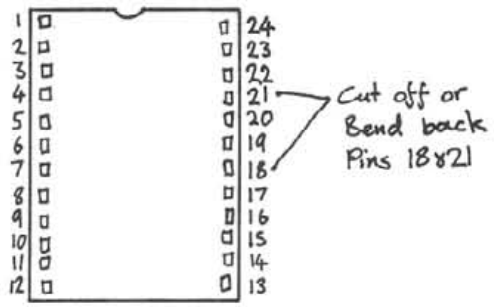
Then take a 20mm length of wire and bend it into a 'U' shape. From on top of the socket, insert the two arms of the 'U' into locations 21 and 24 to a depth of about 4mm, and then gently bend over the wire (taking care not to pull out the ends) to lie as in the diagram.

Take a second piece of wire 28mm long, and bend over 4mm at each end to make a 'staple' shape. Insert the two prongs into locations 12 and 18 of the socket, again from the top.

Now take the 2716 EPROM and (taking care not to disturb the two pieces of wire or bend the pins of either the socket or the EPROM) insert the EPROM into the socket, thus firmly securing the lengths of wire in position.

The EPROM should remain permanently fitted into the socket, and this unit can now be inserted into the Motherboard in place of an Apple ROM.

You should note section 7 of the attached Application Note, which points out that as the INH function will be disabled as far as these units are concerned, they cannot be used in conjunction with a language card - either ROM or RAM. Whilst this is logically correct, and indeed in practise is true with the Apple Language Card, I have found that I can use these units perfectly satisfactorily on the Motherboard of my ITT 2020 in conjunction with the Microsoft RAMCARD.



PREPARATION OF SOCKET

USER FIRMWARE

APPLE APPLICATION NOTES

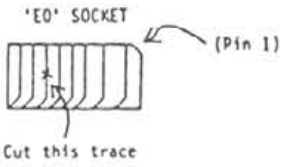
NOTE: The modification described in the following instructions will void the warranty on your APPLE II.

There are times when you may want to create your own firmware for the Apple II. Such firmware can be tailored to your special needs; then installed semi - permanently so that it can be used as conventiently as Apple BASIC or the Monitor.

This application brief details a simple modification which allows your Apple II to accept industry standard 2716 (2Kx8-bit) Eraseable Read - Only - Memories (EPROM's) in sockets "D0" & "D8". These sockets corresoond to menory addresses D000 - DFFF.

EPROM's (2716) are readily available through most semiconductor distributors. Many distributor locations are equipped to program the EPROM's to your specifications, and will so so for a moderate fee.

1. Remove the 'E0' ROM from its socket. On the TOP side of the board, under the 'E0' socket, cut the ROM pin 18 jumper trace. Then reinsert the ROM. This cut will isolate pin 18 or ROM 'D0' & 'D8' from pin 18 of the other ROM. Reinsert the 'E0' ROM when done.



LIFE & IT'S PROBLEMS

BY JOHN SHARP

No, Disgusted of Tunbridge Wells, or Frustrated of Barking, this isn't that sort of a problem page, but is intended as a guide for those of you who have seen the LIFE program on the introductory disk, and are a little puzzled about what it does.

The game of LIFE was invented by the Cambridge mathematician John Horton Conway, and introduced to the world at large by way of the Mathematical Games page run by Martin Gardner in the "Scientific American". The first article was in the October 1970 edition, with a follow up in the November and later editions, notably February 1971.

The game is basically a simulation game, which mimics the process of birth, growth, and death inherent in all life processes. There are certain rules, the "laws of the universe", which dictate how a particular organism exists, and how two or more react with one another.

The rules are:-

1.Survivals. Every cell with two or three neighbouring cells survives to the next generation.

2.Deaths. Each cell with four or more neighbours dies (i.e. is removed) from over population. A cell with one neighbour, or a lone one dies from isolation.

3.Births. A birth occurs when an empty cell is adjacent to just three neighbours. (i.e. it fills on the next generation.)

As a start I suggest you get the program running and follow through these simple shapes to either life, death or immortality.

The program first pokes in a machine code routine to calculate the life processes. Then you will be given some basic instructions. Press return to go to the GRaphics page where you will be requested to enter the X and Y coordinates of the cells to be inserted. Try the following values:-

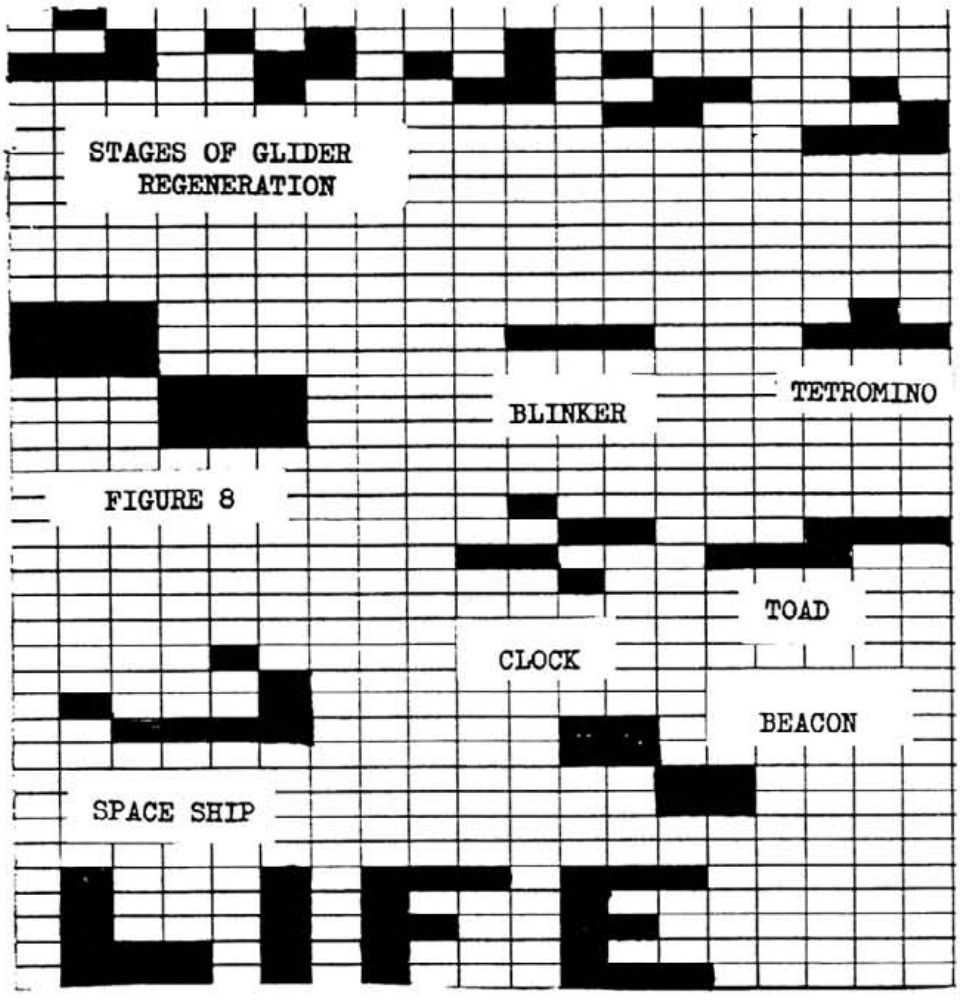
10,10 11,10 12,10 11,9

These are the points for your shape. If you now put in any negative value for X, and then any positive value for Y, the life process will begin. This particular shape will go through a succession of generations, until it reaches a state of oscillation. Some standard life forms have been given names. The form you will end up with here is called the "traffic light".

It is not always the case that a vibrating life form survives. Put in the block:-

10,10 10,11 11,10 11,11

followed by a negative X and positive Y to start the life cycle and it will remain as a stable form.



2. on the UNDERSIDE of the Apple board, cut the traces connecting pin 20 to 21 of ROMs 'D0' & 'D8' only.

3. (Underside) Cut the trace going to pin 18 of ROM 'D8' near the chip. Scrape solder resist off of approximately 1/4 inch of the remaining trace not still connected to pin 18. You may wish to tin it with soder since it will later be soldered to.

4. (Underside) Connect pin 18 of ROM 'D8' to pin 12 of ROM 'E0' (ground).

5. (Underside) Connect pin 18 of ROM 'E0' to the trace which previously went to pin 18 of ROM 'D8' (and which should be pretinned if step 3 was followed).

6. (Underside) Connect pin 21 of ROM 'D8' to pin 21 of ROM 'D0'. Then connect both of these to pin 24 of either ROM (VCC).

7. Note that the INH control function (pin 32 on the Apple I/O BUS connectors) will not disable the 2716 EPROMs in the 'D0' & 'D8' ROM slots, since pin 21 is a power supply pin and not a chip select input on the EPROMs.

8. EPROM (2716) devices may now be used in sockets 'D0' & 'D8'

NOTE: IF YOU MAKE THIS MODIFICATION, DO NOT INSTALL A ROM CARD OR THE LANGUAGE SYSTEM BEFORE REMOVING THE DEVICES IN SOCKETS 'D0 & D8'.

There are some remarkable forms that move across the screen as if they are crawling animals. One such is the "glider" which is formed from the set of points:-
5,5 6,6 6,7 5,7 4,7
Set the glider up and watch it move down the screen.
The low res screen has blocks which are twice as wide as they are high, which confuses the issue slightly if you try to match the diagrams in other publications exactly as you see them; you must allow for the difference in the vertical and horizontal size of the blocks. The program is very fast, and can be slowed down by altering the position of paddle (0). The changes in colour are more obvious then. They show which are the cells being born and which are dying.
The program has the added facility for creating random patterns and then subjecting these patterns to the LIFE process. To try this out, type a positive value for X and then a negative value for Y. You will then be asked for the limits for X and then the limits for Y. This is the size of block in which the random cells will be placed. The values must of course in all cases be within the correct range for the low res screen coordinates, i.e. 0-39.
If the value of X is chosen to be 1 then the area you choose will be completely filled in. Any other value of X in this random case will give a mixture of living and empty cells. Some randomly generated patterns can take 10-15 minutes before they either die completely or reach a stable or oscillating state.
LIFE is a means of exploring with your Apple which can give you hours of enjoyment, even if you only get mesmerised by a randomly generated pattern of cells.

APPLE II UCSD PASCAL REVIEW

BY LEO CROSSFIELD

In using UCSD Pascal I have found a number of frustrating points and indeed a number of desireable traits that make programming a pleasure. At the present time I have not been fortunate enough to explore the area of Turtlegraphics, or file handling; both of which, I am led to believe, have some very nice features.
Before I begin voicing my praises and criticisms, I shall attempt to give a general description of the main three environments in which one must operate in order to utilize the features that Pascal has to offer. For those of you who have never seen UCSD Pascal in action, you will soon realise that the main setup is a totally different environment than, say, that of Basic. For one, you cannot jump out of the language into the Monitor,and also if you are used to testing seperate parts of Basic programs by running a part of the program, you will have to change your method of working. Even the concept of stopping a program and immediately listing it is different. I have found that these, and some other problems exist for those of us that are somewhat adequate in Basic. This can create some frustration, but such changes of thinking are inevitable with any new language and the differences between Pascal and Basic are quite considerable. The major reason is of course the need to compile all text files into p-code before being able to run the program. You therefore need to move between three main environments within the Pascal system. Initially these environments can cause some bewildering difficulties, but these seem to be mainly due to unfamiliarity with the system,and that is soon overcome.
The three main working environments are:
1.THE SCREEN EDITOR.
This editor is not unlike a powerful word processor, and indeed, by setting the environment, can be turned into a word processor.
The editing facilities are somewhat amazing at first if one has only ever used Basic. Insertion, Deletion, Copying the Buffer, and Exchanging Characters are allowed; and of course an 80-column window that is accessed by using Control-C.
All in all the editing facilities are quite impressive. I must admit that I found it most obscure to have to enter different modes to insert text, delete text, and make general cursor moves. You must decide which mode to enter, and at times strange things appear to happen - due of course to a misuse of the system. But, these weird happenings do become less frequent the more one gets to know the system.
To test ones' program, one needs to type "Q" (for Quit) and is then confronted with four options:-
-Update work file
-Exit to command level without updating
-Return to Editor
-Save to file name
Typing a "U" will return you to the outer command level after updating the work file.
The second environment is:-
2.THE COMMAND LEVEL
This allows you to compile any text file on the disk as well as enter Edit or File environments. By typing "R" the system work file will be compiled and eventually run. Needless to say, any syntax error will crash the compiler and

you will be allowed to return to the Edit mode in order to update the file. On returning to the edit mode the cursor will be positioned as near as possible to the error whilst at the top of the screen you will be given an indication as to what the compiler considered had caused the crash. Due to the speed of the compiler it usually overshoots the position in which the error resides.

I found at first that the solutions to the errors were difficult to ascertain, but after a short amount of use one gets to know the main errors. These usually consist of simple mistakes such as a missing ";" on the line above.

The system, I feel, is not at all designed for terminal creativity. It is not unlike mainframes in that you must sit with a pen and a thousand reams of paper and systematically lay out your thoughts and designs, using all the tools of flowcharting, tree structures, etc.

Many Apple user's may at this point think that this is somewhat idealistic but believe me, sitting waiting for a compiler to compile any substantial program that is loaded with half finished logic and syntax is enough to test the patience of the Gods! I know because I have tried.

I can only suggest that you should sit down, get it right, check it and double check it and only when you are absolutely sure type it in.

All disks that are to be used with the language system need to be formatted, and this can be done by Xecuting the formatter code on the Apple3 disk or one can initialise the disk under 3.3 DOS.

To copy any file (or indeed the whole disk) onto the newly formatted disk one needs to enter the third environment:-

3.THE FILER.

This level enables one to housekeep and catalog, and holds many other useful utilities. For instance, by typing "B" you can make a bad block scan of the entire disk, while "X" will examine the range of bad blocks found by the scan, and if you desire will mask them by creating a dummy file in the directory that lives over that area.

There is also a Crunching facility. The system needs, unlike Basic, contiguous sectors, and therefore by removing files and thus leaving areas unused between files one can create a problem of space.

This indeed was my main problem on a one-drive system. It does not constitute too much of a problem until your program has grown to a size of approx. 24 blocks, but I then had to remove the System File to allow enough room to update the system work Text and then compile that text.

If the Text is 24 blocks long then the update procedure places the new work file text next to the old and then deletes the old (leaving a hole of 24 blocks). The compiler then compiles the 24 blocks of text into the 24 block hole. This pair of procedures does not allow you to repeat the sequence due to limited space.

I therefore loaded the File from another source disk (due to the fact that it had been removed from my main system disk to give me more room). Secondly I have to remove the system work code. Thirdly I crunch the disk contents, and finally I leave the Filer and enter the editor to update the workfile.

I'm sure this sounds much worse than it is to do - I found it more difficult to write than to perform, but nonetheless it is a problem. Pascal does need system files - as do most other languages - and the real answer is twin drives. Then the system works fine - one drive houses the system disk while the other drive houses your working Texts and codes.

You may well now be asking "What's the point?". Well indeed compilers are a hassle, but they just require different thinking from the user. Pascal is undoubtedly a powerful language. It is said to be rising in popularity, and I am pleased that I can run it. I have enjoyed using Pascal - it's syntax is friendly and it's structure commonsensical.

Let me explain the fundamentals of Pascal concepts. It is a block structure language which consists of three main elements:-

1.Procedures
2.Functions
3.The main program

Functions are mathematical functions which can be called as easily as calling the function name. Likewise procedures are small programs that are complete within themselves. Each procedure has a function, i.e. a program header could be called "Procedure Header;". The procedure is then defined underneath the procedure name.

Simple rules of syntax govern the structure, but in essence if you want the Header to be called you just write (within the main program which is written AFTER all the procedures) the words "Header;", and the procedure called Header will be put into operation.

Each procedure is a block and must start with the word BEGIN and must finish with the word END. Everything between these two words is the procedure.

The main program is also contained within these parantheses, but is kept to a simple structure without too many control loops or procedure calls. That is to say, if you wish to call a procedure, your program should be structured so that it is called by the previous procedure rather than the main program.

The flow of control can have some nice structures embedded:-

```
1.REPEAT
    BEGIN
    ---
    ---
    END
  UNTIL<CONDITION>
2.WHILE<CONDITION>DO
    BEGIN
    ---
    ---
    END
3.FOR I:= 1 TO 10
    BEGIN
    ---
    ---
    END
```

(Similar to Basic For/Next loop)

```
4.IF<CONDITION>
    THEN--------
    ELSE
```

And of course Pascal also incorporates CASE statements that are in essence cascading IF,THEN,ELSE statements.

One important element in Pascal is its ability to contain Nested procedures, Loops, Case statements, WHILE / DO statements. As far as I know there are no limits to how far one can go in nesting statements and flow control.

Pascal offers procedures within procedures within procedures, and such a structure is obviously of immense importance if one wishes to work with recursion. Recursion is perhaps Pascals' strongest selling point along with its simplicity in tracing flow and being able to see exactly what the programmer had in mind. Of course, being able to compile to code is an important factor.

In writing this article I must admit I have become somewhat carried away, but I have tried to put forward the facts that are apparent when using Pascal. It must be admitted that you definitely need to change your way of thinking if your only previous language has been Basic. But don't get me wrong - I'm not trying in any way to pull down Basic. I feel that both languages have their uses, and must admit that I now realise that Basic gave me a false sense of security, while Pascal woke me up to the new world. I feel it has changed my way of thinking,and in so doing made me better capable of writing programs in Basic and, eventually in other languages as well.

PASCAL ON THE ITT 2020

By Richard Teed

The ITT is almost completely compatible with Pascal, but there are a couple of problems which are as follows:-

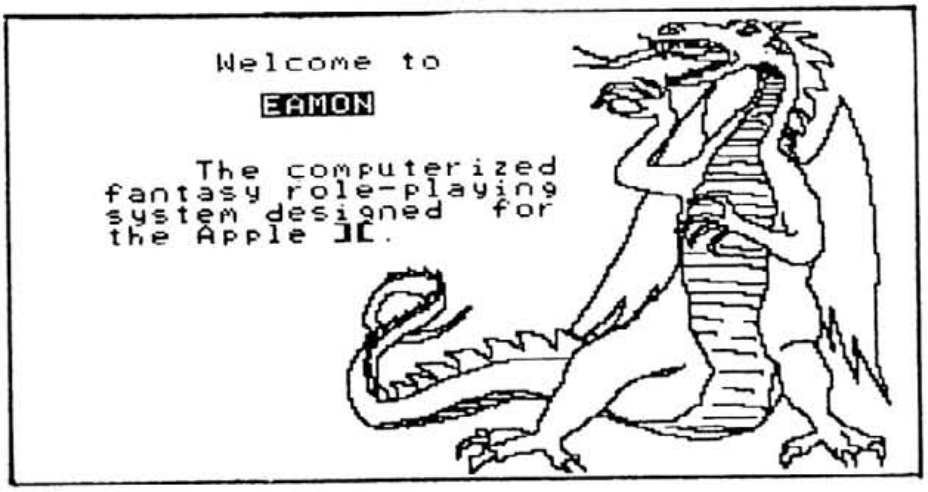
1.When you use high res graphics you will find the Auto Start ROM leaves a lot of the BIT nines' on; so you have white vertical lines down the screen. To get rid of the white lines all you have to do is an HGR from PALSOFT before you start PASCAL. Problem solved?. Yes but there's a catch: when you go into the editor and type control K you don't get an open square bracket. It is now almost impossible to edit a program. To get round this all you have to do is switch off and start again.

2.High res graphics have the usual problems of missing eighth and ninth BITS and BIT eights in high number colours where you don't want them.

EAMON

BASUG

PLAYER'S MANUAL



WELCOME TO THE WONDERFUL WORLD OF EAMON

by Donald Brown

EAMON is a computerized version of what are called "Fantasy Role-Playing Games." When you enter the universe of one of these games, you are no longer John (or Jane) Smith, mild-mannered computer hobbyist. Instead, you become a character in a land of adventure, doing almost anything you want to.

In the land of Eamon, you will be a member of the select Free Adventurers Guild, which is made up of hardy individuals like yourself who want to live by your wits, defeating horrible monsters and finding glorious treasures. (For those of you who want a more calm life, you will have to wait for the game "ADVENTURERS IN THE LAND OF THE CERTIFIED PUBLIC ACCOUNTANTS".)

Unlike most games, there is no single set goal for you to achieve, no way to 'win' the game. Instead, in Eamon, you have a lasting goal to both better yourself and also get rich. If you set for yourself another goal (do good to all princesses, kill all evil wizards, that sort of thing), you may also work towards it in your quests.

To run the adventures of Eamon, you need an Apple II, one disk drive, and at least 32K of memory. (Some scenarios may require 48K.) You do not need this manual (although it does help keep you informed, and informed adventurers survive longer!). The one thing you must possess for Eamon is a large dose of imagination.

ACKNOWLEDGEMENTS: The full list of people who deserve mentioning here is too long to give, but a few are—Bill Fesselmeyer, for introducing me to FRP games, my father for introducing me to the Apple, the many good friends who have play-tested this for me, to all the creators of the games I have played and to the writers such as Tolkein, Leiber, and Niven who have given me so many ideas. And, last but not least, to the talented people of Ann Arbor, Michigan who designed that lovely Dragon Picture.

The basic system of EAMON was created and developed by Donald Brown. The individual adventures were created by various people. Non-commercial distribution is encouraged.

Far away, at the dead center of the Milky Way, is the planet of Eamon. It doesn't orbit any suns—all of the suns orbit it. The shifting pulls of all of these great bodies bring strange forces to bear upon this planet; twisting light, tides, even the laws of science itself! Strange things happen there, and the citizens of Eamon must always be adaptable, for things are rarely what they seem, and even more rarely what they were yesterday!

You are a citizen of this weird world. You are a free man (or woman) out to seek your fortune in this world of shifting laws and time. You will usually find yourself fighting terrible monsters such as Orcs, Trolls, and Dragons to get their treasure. However, at times you may find yourself fighting such varied opponents as Billy the Kid and Darth Vader! Anything can happen, anything at all.

EAMON is a fantasy role-playing game. This means that the computer will generate a character for you and you will pretend to be that person. You will command your character into fierce battle, where hopefully he/she will emerge victorious and wealthy.

Obviously, not all characters are equal in ability. Three numbers (called attributes) describe various parts of your physical condition. You also will have various abilities with weapons, which will increase as you gain experience with them, and learn how to better use them. Additionally, you will be able to learn some powerful magic spells. (Of course, you will have to be taught these spells, and the teacher will charge you for the job!)

EAMON is usually non-sexist—there is full room for both male and female adventurers. However, for simplicity's sake, an adventurer will usually be referred to as 'he'—please understand that it refers to 'she' adventurers also.

CHARACTER ATTRIBUTES

As mentioned earlier, three numbers describe the basic 'working material' of your character. They are all gotten by selecting three random numbers from one to eight and summing them, thus the numbers can range from three to twenty-four, with more numbers around twelve to fifteen. (By the way, this is called 'three die eight' or written as '3D8'. This terminology comes from older role-playing games where you roll strange dice, and means roll three eight-sided dice and add). The three attributes are

HARDINESS, AGILITY, and CHARISMA. Their descriptions and effects are given below—

HARDINESS

Your character's hardiness has two major effects. The most important is that your hardiness is the number of points of damage that your body can withstand before you die. In other words, assume Hedric the Horrible is fighting a Troll. Hedric has an HD (hardiness) of 13. The Troll swings his Battle axe (as described later in the COMBAT section of the manual) and hits Hedric for 10 points of damage. This brings Hedric down to three more points of damage before death—if the Troll can hit Hedric again and do more than two points of damage (before Hedric can go home and heal himself, or use some magic to heal), Hedric will die!

The other effect of hardiness is the total weight that you can carry. The standard measure of weight on Eamon is the Grond, which can be split into ten Dos. You can carry up to ten times your hardiness. Therefore, Hedric can carry up to 130 Gronds (or 1300 Dos). Note that weight-carrying ability is based on the characters base hardiness, not the number of hits he has left. In his unpleasant encounter with the Troll, Hedric can still carry 130 Gronds, even though he only has three hits left before death.

As with all three basic attributes, a character's hardiness is not normally changed. (Unusual magic items or spells might change them). Thus, a player who starts life as a 90-pound weakling will remain one until he dies.

AGILITY

The second basic ability is the player's agility (abbreviated 'AG'). Agility's major effect is in combat—a player with high Agility is more likely to hit an opponent. Agility may also be useful for avoiding special traps (like falling down a mine shaft) or other special occurances.

CHARISMA

The last basic attribute for the player is his charisma (abbreviated 'CH'). Charisma is mostly a measure of physical attractiveness, although it also includes such things as a forceful manner, pleasant speaking voice, and anything else that makes people look at you and say, "Gee, what a nice guy!" (or girl). In some ways, charisma may be the most important attribute, at least for the beginning character. The first major effect of charisma is on the prices you'll have to pay for goods and services (or the prices people will pay you). Obviously, if somebody likes you, he will give you a better price than if you disgust him.

The second effect of charisma is on how citizens of Eamon (generically called monsters) will react to you. Not all monsters are bad—you can sometimes make friends with a few of them, and their assistance may make the difference between life and death! Your charisma will affect the likeliness of their liking you—subtract 10 from your charisma, multiply the difference by 2, and the result adjusts the percentage chance of a favorable reaction from the monster—if there was any chance at all! EXAMPLE! The Mad Hermit of the Beginner's Cave has a 50% friendliness rating, which means that Joe Normal with a charisma of 10 will get make friends with the Hermit one-half of the time. However, old Hedric the Horrible with his charisma of 5 has only a 40% chance of making friends (5-10=-5, -5x2 =-10). On the other hand, Lovable Linda with her charisma of 24 has a 78% chance of making friends. Unfortunately a rat with a friendliness rating of 0 will never make friends, be it with Joe Normal, Hedic, or Lovable Linda.

COMBAT

Being a rough and violent world, combat is the most important aspect of Eamon. In most adventures, combat is taken care of on a blow-by-blow method —every player or monster in turn uses his weapon(s) on one enemy, the effects are calculated, and then applied.

Every time that a player or monster attempts to strike someone else, there is a percentage chance of success. The computer will generate a number from 1 to 100, and if the number is less than the chance to hit, the blow did strike.

Several factors determine just what that chance to hit is. If a player has no armour on, there are three factors—the player's agility, his ability with that weapon, and the quality of the weapon (also called the complexity).

Roughly speaking, all weapons in the world of Eamon can be divided into five types—axes, bows (this includes all thrown weapons and guns), clubs (or any blunt weapons), spears (or other pole weapons), and swords. Every player has what are called 'weapon expertises' for each class. All players start at the same levels: 5% for axes, -10% for bows, 20% for clubs, 10% for spears and 0% for swords. (These numbers are to reflect the fact that somebody who doesn't know what he's doing is more likely to hit with a club than with an arrow.) Your chance of hitting your target is equal to twice your agility plus your ability for the weapon you are using, plus the complexity of the weapon you are using. For example, our old friend Hedric has an agility of 20 and is using a fair quality sword (with a complexity of 0%). Since he is a starting character, he has a sword ability of 0%. Thus his chance of hitting is 40+0+0 or 40%.

Weapon expertises can be increased through use in combat. The scheme goes as follows: Assume Hedric is fighting his troll and scores a successful hit. The question now is, did Hedric learn anything about how to use his weapon better? Well, it just so happens that his chance to learn is his chance to have missed. Thus, 60% of the time Hedric will learn from his blow. If he does, his sword expertise will go up by 2%. Thus, next time his chance of hitting will be 42%. (Notice that his chance of learning on the next successful blow is only 58%.)

Well, Hedric somehow made it out alive from his Troll battle, and has brought his sword expertise up to 12%. He then wants to take his booty and new knowledge and get a better weapon. If Hedric goes and buys a new sword-like weapon, such as a rapier which has a weapon complexity of 15%, his chance of hitting with it will be 40+12+15 or 67%. However, if he decides to switch weapons and get a Battle axe with a complexity of 15%, his chance with that will be 40+5+15 or 60%—his experience with swords will not help him with his axe.

If an attacker is wearing armour, his chance of hitting may be reduced. After all, one just isn't as agile when one is fighting from within a tin can! A player may carry a shield, which will lower the chance to hit by 5%, and may also wear either leather armour (lowers chance by 10%), chain mail (20%), or plate armour (60%!). However, these numbers are "worst cases". A player becomes used to the constricting effect of wearing armour, and builds an armour expertise (called AE). It is built the same way that weapon expertise is increased—every time a successful blow is landed and the effect of armour is bigger than the player's AE, a check is made on the chance to miss and that is the chance of the armour expertise going up by 2%. Thus a successful blow may increase the chance to hit by 4%. Armour expertise is carried over from each type of armour. Thus if you've brought your AE up to 10% while in leather armour and you go to chain, your chance to hit will only drop by 10%, not 20%. However, the effect of armour expertise can never increase the chance to hit—if the AE is 32% and you go to leather armour, the net effect will be 0, not adding 22%.

In addition to agility, weapon expertise, weapon complexity, and armour, there may be magical or other extraordinary forces at work that will affect the chance of hitting.

When a blow hits, a random amount of damage is done to the target. This amount of damage is based on the weapon and will be given in 'N D N' format. (Remember 3D8 for the three basic attributes?) This base number of damage is usually lowered by the armour worn by the defender—leather armour and shield each take one point of damage, chain takes 2, and plate armour takes 5 points of damage away from that taken on the body (all effects are cumulative and magical devices may act as armour).

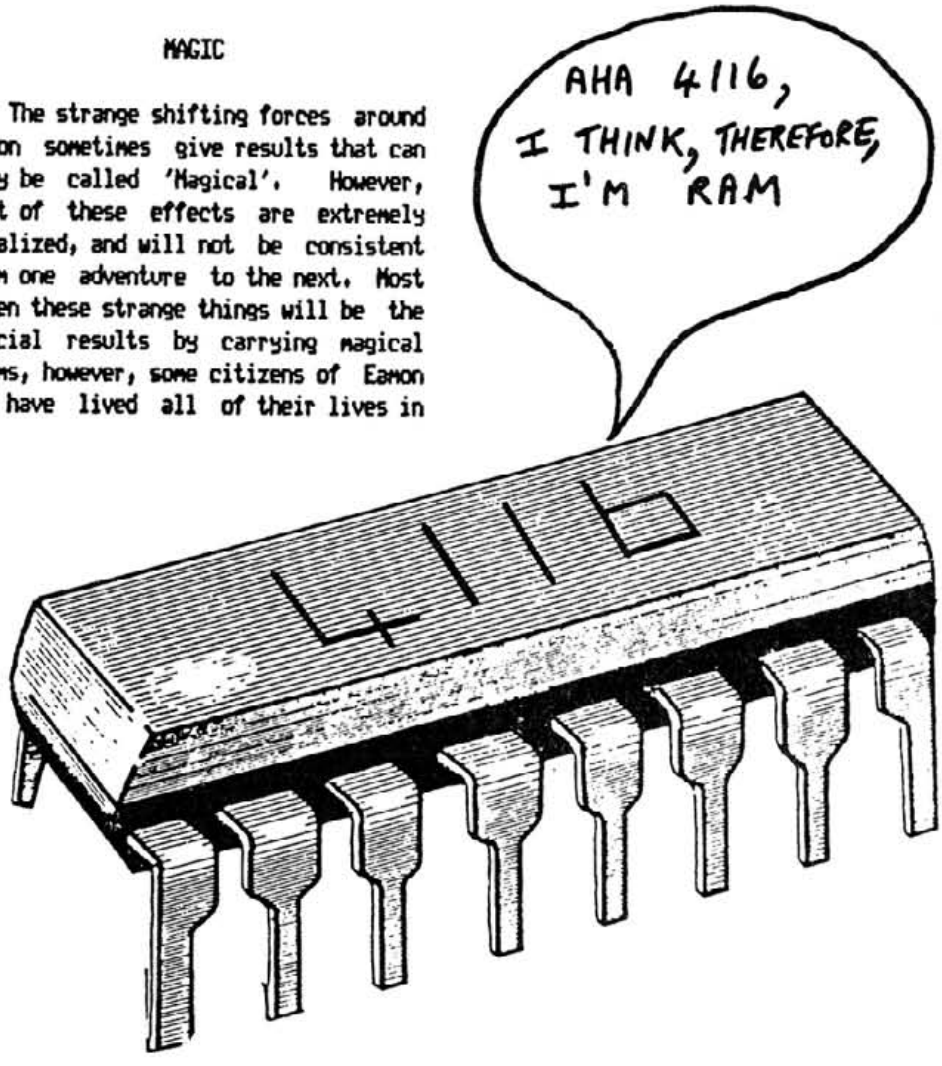
That, of course, is what usually happens. However, due to flashes of good luck or clumbsiness weird things can happen. About 5% of the time an attacker will get what is called a 'critical hit'. That will get one of the following results (each result is followed by the percentage chance of its occurance): Ignore armour (50%), three-halves normal damage (35%), twice normal damage (10%), triple normal damage (4%), or automatic kill (1%).

About 4% of the time the attacker will fumble with his weapon. It will have one of the following effects: Recover from fumble without any other effect (35%), Drop weapon (40%, if the attacker is using built-in weapons such as claws, the attacker simply recovers instead), Break weapon (20%, with a 10% chance of hitting oneself at the same time), Hit self normally (4%), and Hit self with double damage, ignoring armour (1%).

MAGIC

The strange shifting forces around Eamon sometimes give results that can only be called 'Magical'. However, most of these effects are extremely localized, and will not be consistent from one adventure to the next. Most often these strange things will be the special results by carrying magical items, however, some citizens of Eamon who have lived all of their lives in one location may have learned how to control the forces around them.

There are, however, four spells that work almost everywhere. Anyone can be taught these spells without too much difficulty (if you can find a wizard who will teach them to you!). When you learn a spell, you will start with a random ability in it from 25 to 75% (you will not know what your ability is). As with combat experience, this can be increased every time you successful cast the spell—If a random percentage roll is less than your chance to not have cast it, your ability will go up by 2%. However, there is a catch in casting spells—due to the tiring effects of sending all this power through your body, every time that you attempt to cast a spell REGARDLESS OF WHETHER OR NOT THE ATTEMPT WAS SUCCESSFUL your chance for the rest of the adventure is halved. Thus, old Hedric who knows a spell with a 200% ability will always cast it the first time. His second try will also always work (100% of the time). His third try will only work half (50%) of



the time. On the fourth try the chance is down to 25%, fifth try 12%, and sixth only 6%. Fortunately for Hedric, however, if you know a spell your chance of successfully casting it can never be less than 5%, so Hedric can use his spell for the rest of the adventure at the same odds.

The four basic spells are:

BLAST

This spell sends a magical burning arrow at your opponent. Armour will absorb damage from it, but if the spell is successfully cast it will always hit its target, regardless of the range. However, the Blast spell only works on living (or at least animate) objects and the targets must be seen by the person casting it. The arrow will do 1 D 6 of damage (a random number from one to six).

HEAL

The Heal spell removes hits from the body of the person casting. It will cure 1 D 10 hits, but never past 0. (Hedric, having taken five hits, casts a Heal spell on himself successfully. He got a good casting this time—would have cured 8 hits normally, however it only cures five hits one him, bringing Hedric back up to perfect condition).

SPEED

This powerful spell will double the caster's agility for from 10 to 39 turns. You will know when you have cast the spell successfully, however you will not be told when it wears off. If you successfully cast a Speed spell while one is already in effect on you, the new spell will reset the time for you—it will not have the effect of quadrupling your agility. Obviously, when you cast the Speed spell your chance of hitting goes up accordingly (Hedric had a 56% chance of hitting with some weapon before casting the spell, with 40% of that because of his 20 agility. When he casts the Speed spell on himself, his chance will increase by 40% again, giving him a 96% chance of hitting).

POWER

The Power spell may well be the most powerful spell available to you, and certainly the most uncertain. It has no set effect, it's a call to the Gods saying "Hey, do something!". What they do will certainly differ from place to place, and may even differ from one moment to the next! It could kill all of your enemies, teleport you randomly somewhere else in the place you are exploring, cause an earthquake that buries you and your opponents alive, or anything else you can and cannot think of.

For all of these spells, it should be pointed out that this is the way they >> USUALLY << work out. In some obscoure sections of the world spells may not drop in ability every time you use them, in other places spells may not work at all!

RELATING WITH CITIZENS

There are two places you will be encountering other people of Eamon, on your adventures and at the Main Hall of the Guild of Free Adventurers.

At the Main Hall, you will be able to communicate with the various people there and do business. However, they will not do you any real favors (except possibly giving you good prices on things if they like you), and you will not be permitted to fight with anybody there. Essentially, they will be businessmen and women, out to relieve you of some of your gold while helping outfit you to go get more.

On the other hand, during your adventures outside of the Main Hall, you will not be able to communicate with most of the people you find. Additionally, they will usually be rather simple-minded—when meeting you they will decide if they like you. If they do like you, they will follow you around and fight on your side during any battles. If they don't like you, they will try to kill you. These people are rather set in their ways—once they make up their mind about you they will usually keep with their decisions, unless you do something nasty such as attack a friend, or you do something especially nice, such as healing an enemy.

However, just because they do or do not like you does not mean that they will always fight to the bitter end. Some people or things you encounter will be less courageous (or smarter) than others and will run from what they view as a losing battle—both your enemies and your friends. When someone retreats they usually kick up a cloud of dust so you cannot see which way they ran, although they will always only run out of exits that are really there, and you can usually follow them.

Once again, though, note that all of the statements above were prefaced by the word 'usually'. In some parts of the world you may be able to work quite well with others, give orders, get ideas, even play games with them. As always, the key word is flexibility.

HOW TO REALLY AND ACTUALLY PLAY EAMON

(Never thought we'd get here, did you?)

To actually run EAMON on the Apple II, you must first 'bootup' on the diskette marked "THE WONDERFUL WORLD OF EAMON" or simply "EAMON MASTER DISK." It must be in slot six, drive one—Eamon doesn't know how to handle any others yet.

After you are shown the title page (which you can break out of early by hitting the "ESC" key), you will be almost ready to enter the Main Hall. Simply follow directions (for the sake of your mothers, if nothing else!). If you are new to Eamon (or your character was killed the last time he went out), you will be directed to the man in charge of new adventurers. He will show you what the attributes of your new character are, and let you read some instructions that are stored on the disk. If you have this manual, you don't need to read his instructions. Finally, you will be sent to the Main Hall, where all old adventurers go immediately from the Irishman.

The Main Hall will serve as your headquarters. You can buy spells there, as well as weaponry and armour, you can 'check out' yourself and all your attributes and abilities. You can also keep some money with the banker there. He gives no interest, but money in the bank is safe if you're robbed on an adventurer. (Of course, you can't use it to ransom yourself out of a sticky situation, either!).

GOING ON AN ADVENTURE

Of course, the main purpose of the Main Hall is as a place to leave from to go on adventures. Most of your exploits will be exploring caves and old ruins, doing similar things as in the popular Adventure games. However, Eamon is wide enough to also have you go to casinos and gamble your money away, raise an army to fight invaders (both from other countries and from space!), or do just about any other activity you can think of.

Only one Eamon adventure will be stored on a diskette. To go on an adventure, work from the Main Hall as directed, inserting the diskette with the new adventure into the disk drive at slot six, drive one at the proper time. From then on, you're on your own. (Notice: characters who do not return from adventures are considered dead. Thus, turning off the computer in the midst of an adventure or halting it by Ctrl-C or 'RESET' merely commits suicide).

To help your character get some gold to equip himself properly and gather a little bit of experience, one adventure is included on the diskette—The Beginners Cave. It's a gentle little romp through a set of caves underground. I strongly advise that you do send your new character through this first. If he can't survive this, there's no point in going out to the dangerous places. (For more information on The Beginners Cave, see the enclosed sheet).

BUYING WEAPONS AND ARMOUR

You will have 200 gold pieces when you start a character, and hopefully more after your adventures. One of the most important things for you to do with this gold is to buy weapons and armour. Additionally, you may sometimes want to sell a weapon, be it because you have no need of it or because you have reached the legal limit on weapon ownership of four.

Well, Marcos Cavelli owns a small weaponry store in the Main Hall that will do this for you. Marcos carries five standard weapons—an axe, which does 1D6 of damage and has a base price of 25 gold pieces, a bow which does 1D6 and has a base price of 40, a mace which does 1D4 and has a base price of 20, a spear which does 1D5 and has a base price of 25, and a sword which does 1D8 and has a base price of 50. For each weapon Marcos sells three grades of quality—poor (with a weapon complexity of -10%, but only half the base price), medium (with a weapon complexity of 0, at normal price), and good (with a weapon complexity of 10%, at double the base price). Furthermore, the price you are given can vary from one-third to three times the normal price, depending upon how your charisma and how Marcos feels about you.

Marcos will also buy old weapons. If it's of a type that he doesn't sell, Marcos will pay an average of 100 gold pieces for a weapon. If it is a weapon from his stock, he will pay around 1/4 the normal price.

Marcos's base prices for armour are 50 gold pieces for a shield, 100 for leather, 200 for chain mail, and 500 for plate armour. He will also give you a trade-in of your old armour at its old price, subject to adjustment for the way he feels about you.

Marcos's credit terms, like all of the businesses in the Hall, are very simple—none.

BUYING SPELLS

Hokas Tokas, the local wizard in the Main Hall, is willing to teach anybody spells for a price. His base prices for spells are: Power (100 gold pieces), Heal (1000 gold pieces), Blast (3000 gold pieces), and Speed (5000 gold pieces). As with Marcos, Hokas will adjust his prices for how much he likes you, but he will never give credit. But, however he may grumble, he is a nice fellow and will never do anything to you if you try to buy a spell you can't afford, or try to buy a spell twice.

THE BANKER

Shylock McFenney, the local banker, will open up an account with anybody. He is absolutely trustworthy with the funds you leave in his care, although he does not give interest, nor does he make loans. (He makes enough money from adventurers who deposit money with him and never come back.)

EXAMINING YOURSELF

Unlike most things at the Hall, it does not cost you anything to examine your attributes. It is generally a good idea to examine your attributes last thing before leaving to go on an adventure, and write them down—you cannot examine yourself in the midst of an adventure!

LEAVING THE UNIVERSE

This is simply ending the game. However, your character is stored on the diskette, so he or she can be called up again the next time you play. You should only leave the system this way—otherwise some disk files may be destroyed, and your character will be trapped forever in the horrible bit bucket!

That's really about all there is to say about playing Eamon. Of course, the best way to learn is by starting up a character and running him through a few adventures. One thing I would warn you about—do not get too attached to any character. Unfortunately, while wealth and expertise come rather quickly in this world, so does death.

I am very interested in any and all comments and suggestions on Eamon. I am particularly interested in getting copies of adventures that other people create for Eamon. If you want to build your own adventures, all of the tools I used in creating the Beginner's Cave should be on the master diskette. Feel free to list and examine them to help build your own. However, do not at all be constrained by them, the theme of Eamon is do what you want to with it. Eamon hereby officially belongs to the people who play games on computers, all I ask is that you enjoy it.

THE BEGINNER'S CAVE

The Beginner's Cave has been set up by the Warlord as a service to all Free Adventurers, giving them a chance to try their skills in a not-too-dangerous setting. Let us all toast the Warlord for restocking the cave daily!

Only beginners are permitted in the Cave. A beginner is someone who has no armour expertise and who still has all of the starting levels of weapon expertise. You are permitted to carry in only one weapon and any armour you wish. You will not need torches as there is sufficient light in all parts of the Cave. A Knight Marshall (William Misslefire) is on duty to be sure that you do not break the rules (and to keep you from doing something really stupid, like entering the cave without any weapon at all!).

Once you are in the Cave, you will give commands by entering verbs and subjects, such as "GET STONE". If you use a verb that the computer doesn't understand, all verbs will be listed. You must be very exact and use the words that the computer knows. For example, if you are carrying a DEAD MONKEY and you say DROP MONKEY the computer will not understand.(Sometimes the computer does recognize more than one word for an object, though). If you want to repeat the last command given, simply hit 'return' when asked for your next command.

A few commands you should know about:

N, S, E, W, U, D, NORTH, SOUTH, EAST, WEST, UP, and DOWN all will move you in the direction given.

INVENTORY or "I" will list all of the items you are currently carrying.

READY brings a weapon into 'ready' mode,meaning that it will be the weapon used in an ATTACK command.

GET picks up an object (not a monster!) from the floor. GET ALL gets all objects there. If you get a weapon and you have no weapon ready, it will ready that weapon automatically.

Other commands are either self-explanatory or they are designed to make you experiment.

To return to the main hall, you must leave the cave (getting to the Cave Entrance) and move North. Once you have done so, Sam Slicker (the local dealer for treasures and booty) will pay you what they are worth (with the price adjusted by your Charisma). You will then be returned to the Main Hall.

Of course, that is only if you survive. If you died, remember that it probably wasn't that great of a character anyway!

NOTE: If you accidentally stop the program while it is running, (such as accidentally hitting 'reset'), you may be able to continue by first getting back into BASIC and then entering:

]POKE 51,0:GOTO 1000

THE MINOTAUR'S LAIR

The method of running the Minotaurs Lair is roughly the same as it was in the Beginners Cave. Of course, the monsters and treasures and rooms are not the same, and some of the verbs that the computer recognizes are different, and the Power spell may have different results, but why quibble?

A major difference now is the fact that you will not know the way out. A hint to adventurers:In cases like this, your top priority should be finding a way out and mapping the dungeon. Also, only a real louse would not at least try to find a good friend if he/she knows one is in here. (Remember, louses do not have high charisma!)

Pippin's Page ~~~~

Edited for younger readers by Vernon Quaintance

Welcome to this second Pippin's Page. You will recall that last month I set you the problem of saving a record of the number of games played, and the number won, whilst clearing all the other variables before starting another game. Only one person has so far sent in an answer.

Thank you very much Alan Sausse (age 11). Your answer is neat and I like the last line very much, it is shorter than other endings which I have seen. You might like to consider what difference, if any, would be made by having a machine code routine loaded at $0300.

I will give you all until the next issue to try your hand at this problem. Alan's solution, and any other good ones will be published in the next issue of Pippin's Page.

Have any of you experimented with the Organ program which is on the BASUG Introductory Disk? I have recently heard that Jennifer Bolton enjoys using this program very much. At only 13 months old, she must be about the youngest **Apple Systems** user! Do any of you use your machine to entertain younger brothers and sisters? If so, please write and let us know what they like to do with the computer.

Last issue I invited you to send in programs which you have written. These do not have to be very elaborate. They can be games, utilities, graphics, teaching, music, or any one of the many other things that can be done on an **Apple**. We know that we use the best small computer available. Let's see some of the uses to which it is being put.

Do you have problems trying to do certain things? Have you ever wished that you knew where the computer put new programs or some other data? Why not use **Pippin's Page** to ask other users about you problems. Remember the old saying that a problem shared is a problem halved. On the other hand, if you think you know a new trick or two which others might like to use, write in and tell them about it.

Would you like to see **Pippin's Page** used partly for descriptions of new programs and add-on bits? Do you want to know how to add bits on to your computer? I will try to use this page as <u>you</u> want it, so please let me know what you want.

Finally, for this time, here is a short program to try on your **Apple**. But before you run it, write down what you think it will do. When you have run it, compare what it did with what you expected. If they are different then go back and work out why you were wrong. You might learn something new.

```
100   GOTO 140
110   N = M + 1 - I
120   PRINT TAB(13 + N); MID$(P$,N,2*I-1)
130   RETURN
140   TEXT : HOME : VTAB 6
150   WAIT -16384,128 : WAIT -16384,192,191
160   L$ = CHR$(80) + CHR$(73) + CHR$(80) + CHR$(80) + CHR$(73) +
      CHR$(78) + CHR$(39) + CHR$(83)
170   M$ = CHR$(32) + CHR$(80) + CHR$(65) + CHR$(71) + CHR$(69)
180   P$ = L$ + M$
190   M = INT(LEN(P$)/2 + 0.5)
200   FOR I = 1 TO M : GOSUB 110 : NEXT
210   FOR I = M - 1 TO 1 STEP -1 : GOSUB 110 : NEXT
220   GET K$
230   VTAB 23
```

Happy Apple Computing.

SUPER PROGRAMS!!

the software house

C — cassette
D — disc

Our list of software is *FREE*. Our illustrated catalogue costs £1 but contains discount vouchers!

APPLE

ATTACK FORCE (C)	A GREAT *NEW* ARCADE-STYLE GRAPHIC GAME FROM COMPUTHINGS	£9
OPERATION APOCALYPSE (D)	(THE *ONLY* COMPUTER GAME THAT ENABLES YOU TO RECREATE WORLD WAR II LAND OPERATIONS)	£30
TORPEDO FIRE (D)	SUPERB 3D GRAPHIC SUBMARINE SIMULATION	£30
CARTELS & CUTTHROATS! (D)	THE FIRST *FUN* BUT ACCURATE BUSINESS SIMULATION GAME	£22
WARP FACTOR (D)	HAS TO BE THE BEST SPACE SIMULATION AVAILABLE!	£30
SPACE EGGS (D)	*SUPERB* COLOUR GRAPHICS– CRACK THE EGGS—SHOOT THE MONSTERS!	£18
SNOGGLE (D)	FANTASTIC GRAPHICS — CHASE THE GHOSTS ROUND THE MAZE	£16
ALIEN TYPHONE (D)	GALAXIANS *BUT* TWICE AS MANY! TWICE AS FAST!	£15
BASIC COMPILER (D)	A SUPERB TOOL. YOUR PROGRAMS BECOME INCREDIBLY FASTER IN EXECUTION!	£142
TABS (D)	*SUPERB* RANGE OF SOFTWARE MODULES — *ALL* THE STANDARD BUSINESS PROGRAMS AND ONLY (CALL IN FOR A DEMONSTRATION!)	£99 + VAT each!
TIME LORD (D)	EXCITING SPACE STRATEGY GAME	£18

HARDWARE

VERSA-WRITER — MAGIC GRAPHICS EASILY!	£125 + VAT
EPSON MX 80FT — MARVELLOUS VALUE PRINTER	£400 + VAT
VIDEO GENIE 16K	£289 + VAT
16K UPGRADE (12 MONTH GUARANTEE!)	£17 inc. VAT
APPLE II FROM STOCK	

apple ITT

MAIL ORDER
146 OXFORD ST.
LONDON W.1.
TEL: 01-637 2108

RETAIL SHOP (MON-SAT 9.30/5.30)
HORSE SHOE YARD
BROOK STREET
LONDON W.1.

NEW: HIRE SERVICE FOR APPLE & VIDEO GENIE!!!

Accutrack Disks . . .

Because data reliability is the important difference in disk construction.

Anatomy of a disk.

Flexible disks are simple information storage devices consisting of a magnetic disk enclosed in a semi-stiff protective jacket. The disk rotates within the jacket while magnetic recording heads on your data or word processing systems "read" or "write" information on the disk's magnetic surface. Since disk operation is simple, it's relatively easy to make one that works. But building in reliability is something else again. It takes specialized technology to build disks that operate flawlessly over an extended period of time.

What counts in disk construction.

Key design objectives for a disk are listed below. How well a disk measures up to these objectives relates directly to the throughput, accuracy and overall costs for your data or word processing system. No disk measures up better than Accutrack.

- The magnetic coating must be precisely formulated and uniformly applied. Imperfections as small as five millionths of an inch cause signal dropouts, data checks and wasted processing time as well as errors.
- The disk surface must be absolutely clean, totally flat and permanently lubricated to prevent excessive head wear with subsequent signal degradation and eventual loss of information. (This is the most critical objective and the one that's most often compromised since poor operating results take a while to show up. It's also the area that most effects the long term reliability of your data.)
- The disk must be free to rotate within its jacket without internal drag to avoid further data checks, excessive processing times and errors.
- The jacket must protect the disk from external contamination and damage. It should also remove microscopic particles of debris from the disk surface before they can damage the disk.

Why you'll never find the best disk bargain in the bargain basement.

While there's little apparent difference between other disks and Accutrak, the performance differences can be substantial. Simply stated, an Accutrack disk is premium priced. But the protection it gives your information; the reliability it provides to your operations; and its substantially longer life make it the best disk buy. After all, the real cost of your operations is constructing and processing the data stored on the disk – not the disk itself. It doesn't make sense to trust that data to anything but the best disk. Accutrack.

BASUG BULK BUYING – 50% off PRICE FOR 10s

£17:50 incl FREE LIBRARY CASE !

(ADD 50p POSTAGE / PACKING)

BRITISH APPLE SYSTEMS USER GROUP

P.O. Box 174, Watford WD2 6NF.